# NL2IBE – Ontology-controlled Transformation of Natural Language into Formalized Engineering Artefacts

Nicolai Schoch
Corporate Research (DECRC)
ABB AG
Ladenburg, Germany
nicolai.schoch@de.abb.com

Mario Hoernicke
Corporate Research (DECRC)
ABB AG
Ladenburg, Germany
mario.hoernicke@de.abb.com

***Abstract*—Looking at Process and Automation Engineering (P&AE) today, for the technically adept engineer, there are many different tools available to support the engineering work from translation of engineering intentions into module and plant descriptions, to definition and parametrization of entire process plant setups, for export to a control system. However, still today, in the very early engineering phases, engineering intentions either need to be entered already in a structured and controlled expert language or require a human expert's manual efforts for translation from unstructured language into formalized representations, in order for thereon-based consistent further processing in the existing tools. This process is time-consuming, fuzzy, and error-prone due to potential misconceptions and ambiguities, even for domain experts. In this work, we therefore present our NL2IBE Tool, which makes use of modern Natural Language Processing in combination with Ontology Mining, and which, based on and controlled by an underlying ontology, allows for the deterministic transformation of natural language intentions into structured and consistent engineering artefacts. We describe the overall tool architecture as well as crucial functionalities and implementation features, followed by an evaluation by the example of a hydrogen generation and CCSU use case. We conclude with a discussion of the proposed tool and give an outlook on future research. (*Abstract*)**

***Keywords—process & automation engineering, intend-based engineering, natural language processing, NLP, generative AI, ontological domain representation.***

## I. Introduction & Motivation

### A. Engineering Context and Problem Description

Modular Automation (MA) Engineering has recently been a trending topic in the industry [1]. Over the last years, there have also been several pilot applications presented, e.g., by ABB, Bayer, Merck and Evonik [2], and modularization is considered an auspicious approach to handle the challenges of changing production environments. At ABB research, a concept for MA has been developed based on so-called Function Modules (FMs) [3], which bring the benefits of MA without requiring a dedicated controller for each module.

In the context of several partly ongoing research activities, different tools have been developed to implement the FM concept and its facilitation along the entire engineering workflow, from early process engineering via automation engineering to control code export onto target control systems (such as ABB's System 800xA). See Fig. 1, for a visualization of the overall process and automation engineering (P&AE) workflow and the respectively most relevant engineering functionalities and engineering artefacts. In the early phases, research mainly focused on "Intention-based Engineering" (IBE) [4] to support the modeling and formalization of engineering intentions into abstract service and module representations, and on "Module Pipeline Generation" (PipeGen) [5] to facilitate the automated generation of module-to-module pipelines based on the semantic description of modules coming from the IBE Tool and of the intended to-be-processed materials. These two tools, as shown in Fig. 1, are followed by the 'Orchestration Designer' Tool, which facilitates module engineering and plant orchestration [6]. Accompanying, the Semantic Facilitation & Integration Layer (SemFIL) [7] guarantees and facilitates the seamless and consistent integration of the above tools into the overall workflow. The underlying SemFIL ontology and associated knowledge graph therefore store the semantic knowledge and data in a machine-readable way and represents the overall domain knowledge in a controlled vocabulary. In total, these applications thus provide solutions for the process and automation engineering stages in the overall P&AE workflow, see again Fig. 1.

However, still today, in the early engineering phases, engineering intentions either need to be entered into the existing engineering tools already in a structured and controlled formal expert language or require a human expert's manual support for 'translation' from unstructured and uncontrolled natural language into formalized (controlled and structured) representations (for thereon-based consistent further processing using the existing above-mentioned tools).

This process however is very time-consuming, fuzzy, and error-prone due to frequent human misconceptions and semantic ambiguities, even for domain experts.

### B. Proposed Solution and Comparison with State of the Art

Therefore, in this work, we present our NL2IBE Tool, which focusses on the very first step in the described workflow, see the red-dashed circle in Fig. 1.

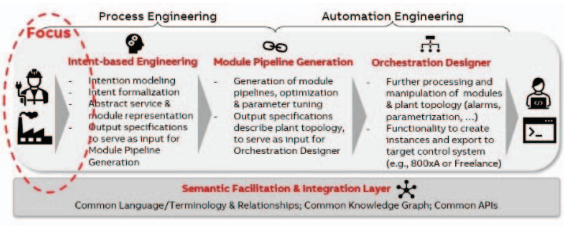


Fig. 1. The overall process and automation engineering workflow along with the existing tools and their respective functionalities and inputs/outputs. The red circle to the left shows the focus of this work, the transformation of natural language into formalized engineering intention artefacts.

979-8-3503-5409-6/24/$31.00 ©2024 IEEE
DOI 10.1109/CAI59869.2024.00182

The NL2IBE Tool supports the engineer in the activity of entering engineering intentions into our system. Without actually requiring him/her to bother about formal and structural correctness of inputs, our tool converts uncontrolled text inputs into structured and SemFIL-consistent ontological representations. NL2IBE therefore makes use of modern Natural Language Processing (NLP) methods. However, it not only translates/formalizes natural language intentions, but also, based on and controlled by the underlying SemFIL ontology, it allows to obtain deterministic transformations of natural language intentions into structured and SemFIL-consistent engineering intention artefacts, see Fig. 2.

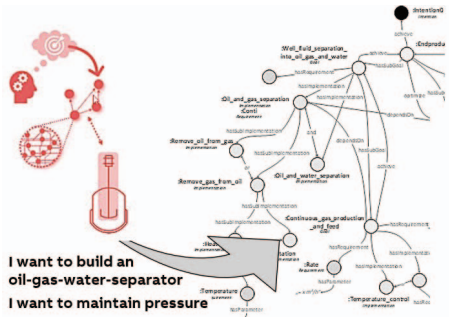


Fig. 2. Our NL2IBE Tool is dedicated to converting natural language engineering intentions into formalized ontological intention representations.

The initial idea was to use standard transformer NNs-based, pre-trained language models, i.e., Large Language Models (LLMs), for the processing and translation of the natural language engineering intentions [8]. Large, instruction-tuned models reportedly can be fine-tuned on I/O schema, so that they adapt, e.g., particularly to a certain domain such as engineering, or to generate output according to a certain schema like JSON or OWL format [9].

Being trained on vast amounts of diverse text data, LLMs can 'understand' complex natural language (e.g., engineering intentions), along with syntax, semantics and context [10]. To achieve this, besides the deep encoder-decoder architecture with several hidden layers, text embeddings are the core feature of the transformer NN architecture, representing words or sentences as numerical *n*-dimensional vectors, which in an abstract way capture their encoded meaning. Hence, formally, natural language input *text* is transformed by a function $f$ into *n*-dimensional embedding vector representations in the latent embedding space $\mathbb{R}^n$:

$$f(text) \rightarrow \mathbb{R}^n. \quad (1)$$

During training and/or fine-tuning, the goal is, among others, to find a function $f$ such that semantically similar text is metrically close in $\mathbb{R}^n$ (e.g., with respect to the cosine similarity metric). Essentially, embeddings thus enable the model to learn abstract relationships between words/sentences and contextualize information [8].

Nevertheless, while LLMs are thus able to understand content and to generate remarkably coherent and contextually relevant responses, they also come with several severe challenges and problems. Among others, these include potential biases in the training data, inaccurate or hallucinated content generation, as well as non-deterministic and non-reproducible resulting output, which is mainly due to the stochastic nature of the underlying NN architecture [11].

Given the fact that LLMs can grasp the abstract relationships between words and sentences, it seems obvious that this abstract understanding of relationships and context should possibly be exploited and be usable also in the context of ontological information modeling and knowledge representation. Several research groups have accordingly investigated on or demonstrated different beneficial aspects of this synergy [12]: Proposed as a roadmap, [13] categorizes three main ways in which LLMs come to interact with ontologies or knowledge graphs (KGs). First, KG-enhanced LLMs, which incorporate KGs during the pre-training and inference phases of LLMs. Second, LLM-augmented KGs, that leverage LLMs for different KG tasks such as embedding, completion, construction, and question answering. And third, mutually synergized and enhanced LLMs and KGs, e.g., for bidirectional reasoning driven by both data and knowledge. The authors in [14] present an NLP-supported method for the structured extension of existing ontologies. The proposed workflow derives classes and relations from text and has a domain expert revise the resulting extended ontology. Similarly, [15] investigates on LLMs' capacities of translating natural language questions into formal database queries and retrieving information from graph databases or KGs. Lastly, while the semantic embedding of KGs has been widely studied and meanwhile been used for statistical analysis tasks across various domains [16], less attention has been paid to developing methods for embedding entire OWL ontologies. Here, OWL2Vec* [17] particularly stands out, as it allows to encode by means of embedding vectors the semantics of single concepts of an OWL ontology by considering its graph structure, lexical information, and logical constructors.

Given this state of the art, and looking at the original problem of transforming natural language engineering intentions into their formalized ontological intention representation counterparts, to the best of the authors' knowledge, to date, no approach has been presented which maps in a controlled and deterministic way natural language to ontological concepts, triples, or even triple combinations from an existing engineering ontology.

In other words, and more generally, pure LLMs-based solutions cannot (yet) *reliably, transparently,* and *controllably* generate formalized output, which is *guaranteed* to be *consistent* with a specific ontology or domain information model representation [11]. This, however, is a requirement for many critical systems applications, particularly in process and automation engineering, where fuzzy 'black box' ML model behavior is not accepted by customers.

## II. Core Idea of our Solution

To cope with the above requirements, our idea was, to add a further '*modality under our control*', namely ontological representations (i.e., concepts, concept-relation-concept triples, and triple combinations), into the *same* embedding space $\mathbb{R}^n$. See Fig. 3, for a high-level visualization of the core idea and workflow of our solution: Using an LLM, we transform, i.e., encode, not only an uncontrolled natural language input sentence $s$ (an arbitrary engineering intention), but also all consistently permitted ontological formulations $c_i$, i.e., concepts and combinations (triples) of concepts and relations, into the *same* joint embedding space. Thereon-based, for the respectively embedded natural language engineering intention vector $\boldsymbol{s}$, we then select the closest ontological representation embedding vector $\boldsymbol{c_i}$ from the set of all ontology-derived embedding vectors $\{\boldsymbol{c_i}\}_{ic\,\mathbb{N}}$.

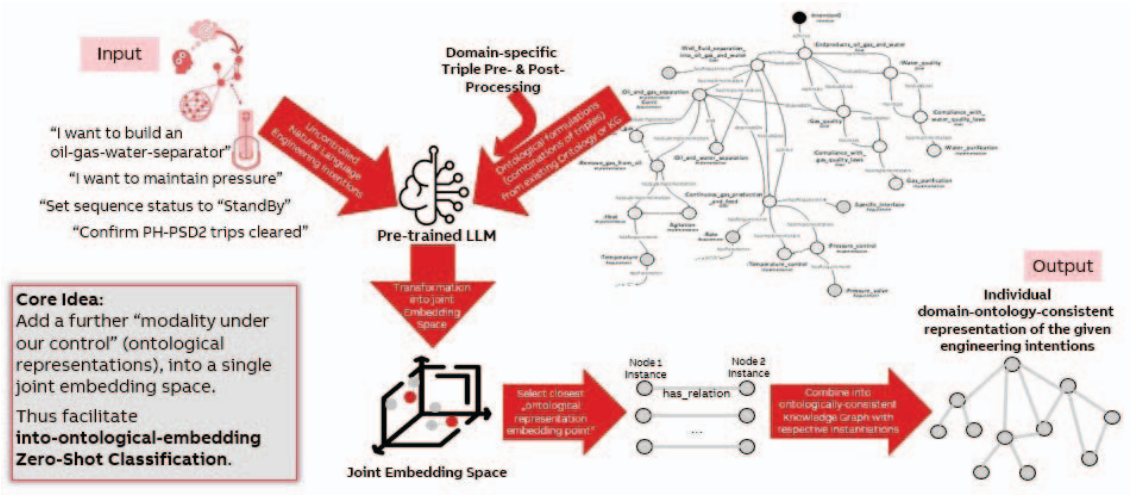


Fig. 3. The core idea underlying our NL2IBE Tool: Adding a further modality under our control (ontological representations) to the joint embedding space, so that new embedding representations of natural language engineering intentions can be analyzed on their proximity and similarity to the well-defined, controlled, and structured set of consistent process and automation engineering formulations.

Figuratively, one could say, we thus facilitate and enable '***into-ontological-embedding Zero-Shot Classification***'.

The above process is repeated for the entire set of natural language engineering intention sentences, which, e.g., in the end constitute altogether the description of a new to-be-engineered process plant.

In mathematical terms, what happens is that we artificially *sparsify* the *continuous* (joint embedding) solution space $\mathbb{R}^n$ in equation (1) into a *discretized* space $\mathbb{R}^{n^*}$ by accepting as final consistent solution of the enhanced transformation $f$ only the deterministic embedding representations $f^*$ of the ontology-derived concepts, triples, and triple combinations:

$$f^*(s) \to \mathbb{R}^{n^*}, \tag{2}$$

where $f^*$ is equivalent to $g \circ f$ with the standard transformation $f$ from (1) followed by identification $g$ of the closest ontological concept or triple $\boldsymbol{c_i}$ in $\mathbb{R}^{n^*}$.

The original transformation $f: A \to \mathbb{R}^n$ (with A the space of arbitrary words and sentences $s$) here is combined with the function $g: \mathbb{R}^n \to \mathbb{R}^{n^*}$, which finds for every embedding vector $\boldsymbol{s}$ in $\mathbb{R}^n$ the respectively closest ontological concept/triple $\boldsymbol{c_i}$ in $\mathbb{R}^{n^*}$, such that

$$f^* = g \circ f: A \to \mathbb{R}^{n^*}, \tag{3}$$

and

$$s \mapsto (g \circ f)(s) \coloneqq g\big(f(s)\big) = f^*(s) = \boldsymbol{c_i}. \tag{4}$$

With this setup, we achieve formalization of uncontrolled and unstructured natural language inputs, by transforming inputs into embedding representations in a joint embedding space with controlled and structured ontological formulations (concept & triple combinations), followed by finding among these the closest fits.

In the next section, we will describe the inner workings and implementation of our solution in more detail.

## III. Our Solution: The NL2IBE Tool

Having motivated and described the high-level core idea of our solution in the above section, we here want to have a closer look at the approach and its implementation.

Therefore, first, we look at the underlying SemFIL and P&AE ontology [7], which we aim our algorithm to produce consistent outputs for, and which hence determines the target output representation. Second, we describe the major steps in our NL2IBE algorithm, followed by a detailed description of its implementation and architecture.

### A. SemFIL and the P&AE Ontology

The overall goal of this work is to obtain from natural language engineering intentions suitable input for the IBE Tool ("Intention-based Engineering") [4], and from there further to Pipeline Generation [5] and Orchestration Designer [6], see again Fig. 1. Alongside entering the overall P&AE Tool Chain we also want to become consistent with SemFIL [7] and with the underlying SemFIL ontology, our controlled & structured P&AE domain representation.

SemFIL was already described in an earlier publication [7], and we will dig deeper in a further future dedicated publication. Here, we only want to give a minimalistic overview on its most important concept groups and relations.

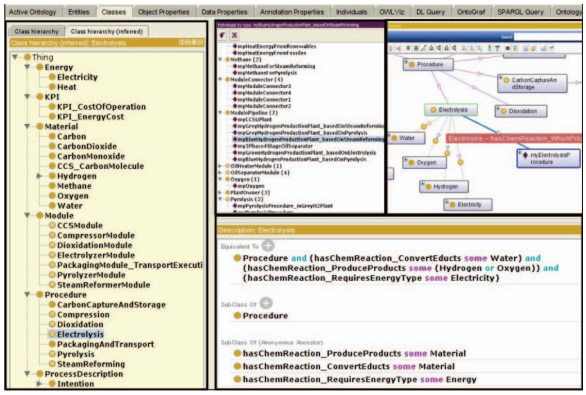


Fig. 4. SemFIL P&AE Ontology (Protégé Screenshot).

To consistently govern the entire P&AE domain and tool chain, it can represent 'abstract data' (which can stand for materials, signals, or energy), and 'data manipulation processes' (e.g., mixing, pyrolysis, or steam reforming). Moreover, it also knows and describes the world of 'modules' and 'services' in the MA context, along with its structures, inputs/outputs, parameters, and KPIs, as well as 'module chains', 'processes', and 'procedures'. Finally, it also formally describes engineering knowledge, as well as the interrelations and dependencies, restrictions and axioms which govern the P&AE domain. See Fig. 4, for a screenshot of our SemFIL P&AE Ontology, taken in the Protégé ontology editor.

This is important since we motivated above, that we aim to transform uncontrolled natural language engineering intentions into their structured ontological formulation equivalents. In order to become consistent with this SemFIL Ontology, we extract from it all concepts and consistently permitted triples (i.e., concepts and instances along with their ontological interrelations), so that we can then find, using NLP, the respectively best suitable ontological representation for a given natural language engineering intention. In the next subsection, we describe this process and its algorithmic implementation.

### *B. Our NL2IBE Solution*

Here, we describe the main steps in our NL2IBE algorithm, followed by a detailed description of its implementation and architecture. We refer again to Fig. 3, for a visualization of the core idea of the NL2IBE solution.

*The Main Steps of the NL2IBE Tool*:

0) Make available for usage a pre-trained LLM to obtain joint embedding representations of both the controlled ontological formulations (i.e., concepts, instances, and triples, and triple combinations) and of the arbitrary uncontrolled natural language engineering intentions input.

1) Use the LLM to obtain the embedding representations of the controlled ontological formulations. Thanks to the underlying ontology, it is known what the ontological formulations mean (semantically) and how they are (inter)-related.

2) Use the same LLM to process (new) natural language engineering intentions input, and accordingly obtain their embedding representations, too.

3) For every such new natural language engineering intention (keyword/formulation) embedding, find the top N ($N \in \mathbb{N}$) nearest (i.e., most metrically similar) known ontological concept(s)/triple(s) embedding(s), along with its/their semantic meaning(s) and interrelations. Above we figuratively described this as into-ontological-embedding Zero-Shot Classification.

4) Rank the top N closest formal ontological representations according to a selected metric (e.g., cosine similarity) or with respect to uncertainties.

5) Allow for similarity- and uncertainty quantification-based human expert assessment and further fine-tuning of the respective transformation results through human expert interaction.

With these main steps being listed, we have implemented our NL2IBE Tool as a Web App. It consists of an HTML- and JavaScript-based frontend part for receiving user inputs and showing processing results, and a Python/Flask-based backend part for performing the ontology-controlled transformation of natural language inputs into ontological engineering intentions.

The **Frontend** functionalities are straightforward; see also Fig. 9 for a screenshot of the frontend of the NL2IBE Tool:

1. The user enters an engineering intention text string into the text input field and clicks 'Submit' to request for NL2IBE-Processing. (See below for what then happens in the NL2IBE Backend).

2. Being returned from NL2IBE-Processing Backend, the user can check out the top 3 best suitable ontological formulations and inspect a visualization which shows in a 2D projection the entered engineering intention and the top 3 nearest ontological representations (based on a Principal Component Analysis-derived projection).

3. The user can select the respectively best fit or raise an ontology extension request (if none of the proposed representations satisfy the actual intention).

4. Upon selection and acceptance of the respectively best fitting ontological representation (i.e., concept/triple/combination), it is sent to the 'History DB' for accumulation, until all intentions are entered.

5. To enter further intentions, repeat steps 1-5 until done.

6. Upon finalization of the engineering intention processing, the user clicks the 'Finalize' button, to have all ontological representations (i.e., concepts, triples, or triples combinations) being combined into an ontologically consistent Knowledge Graph with the respective instantiations. Thus, an individual domain-ontology-consistent representation of the given set of engineering intentions is obtained.

The **Backend** provides the actual core functionalities, building mainly upon *owlready2* library (for ontology-oriented programming) [18] and *transformers* library (for LLM integration & NLP functionalities) [19, 20]:

On Startup of the NL2IBE App, as part of a preparatory step, the LLM is initialized and the ontological embeddings are precomputed once only and made available for subsequent comparison to later entered engineering intentions, see Fig. 5.

```
# Preparatory Steps on Startup of NL2IBE App:

def initialize_llm_and_compute_ontological_embeddings():
    # Extract ontological content (i.e., ontological concepts &
    triples) and process it into NL-equivalent sentences:
    call extract_ontology_content()
        # Extract all concepts & consistent triple combinations
        # from P&AE.owl ontology file.
        returns all_ontology_derived_concepts_and_triples
    call preprocess_ontological_content_2_NL(
            all_ontology_derived_concepts_and_triples)
        # Pre-process (using RegEx) the auto-generated ontology-
        # derived set of (raw) formal sentences/concepts/triples
        # into human-readable ontological-sentence-equivalents
        returns set_of_nl_ontology_equivalents
    # Load LLM (e.g., Transformers Model, e.g., all-MiniLM-L6-v2)
    # & process NL-equivalent sentences into onto_embeddings_list:
    model = load_LLM()
    # Compute embedding representations / embedding vectors:
    onto_embeddings_list = model.encode(
                                set_of_nl_ontology_equivalents)
    returns onto_embeddings_list, dim_embedding_space
```

Fig. 5. Pseudo code of the preparatory steps on startup of the NL2IBE Tool.

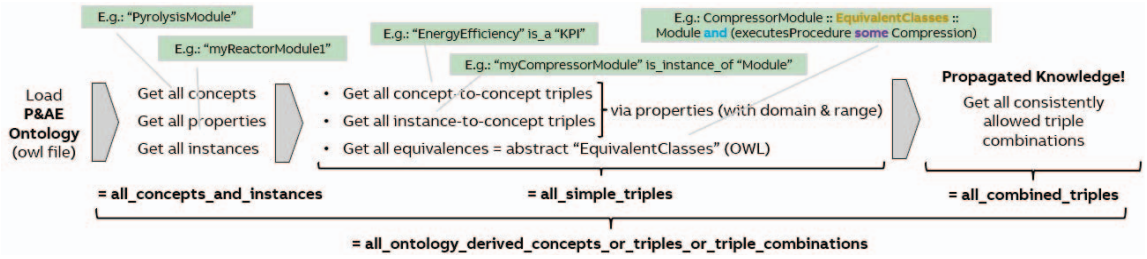


Fig. 6. Visualization of process for the stepwise extraction of the content of P&AE ontology, executed as part of the *extract_ontology_content()* method.

The method *extract_ontology_content()* is to extract all concepts, instances, triples (i.e., concepts and instances, along with interrelations and properties), and consistent triple combinations from the P&AE ontology file, see Fig. 6.

Having extracted, using *owlready2*, from the ontology all concepts and triples, they are further processed, using standard *RegEx* regular expressions, into human-readable ontological sentence equivalents. Thus, for example, a triple '*CompressorModule hasChemReaction_whichTakesAsInput GreenHydrogen*' will be translated into a more-natural language sentence '*Compressor module has chemical reaction which takes as input green hydrogen*'. Only then, this more natural sentence will allow for reasonable subsequent NLP-based processing into embedding representations.

After this once-only-on-startup effort of computing all ontological representation embeddings, the actual NL2IBE core script is called, see Fig. 7. Here, we first artificially de-intentionalize the natural language input, i.e., remove the intention formulation wrapper (e.g., "*I intend to achieve ...*") from the input. This part of the input sentence, and how it affects formal requirements engineering, will be dealt with in a future dedicated work. The actual engineering intention content will then be encoded using the LLM, so we obtain its embedding vector, which can then be compared to all pre-computed ontological formulation embedding vectors, e.g., with respect to the standard *cosine similarity* metric.

```
# Preparatory Steps on Startup of NL2IBE App:

call initialize_llm_and_compute_ontological_embeddings()

# On Execution of NL2IBE App:

def execute_nl2ibe_script(eng_intention_input_string):
    # De-intentionalize NL input sentence
    # (i.e., remove intention formulation wrapper):
    eng_intention_content_in_nl = deintentionize_content(
                                    eng_intention_input_string)
    # Compute embedding representation for eng. intention content:
    eng_intend_embedding = model.encode(
                                  eng_intention_content_in_nl)
    # Compute cosine-similarities between NL input string
    # and all ontological triple combinations:
    onto_embeddings_list_cosine_similarity_values =
                  [pytorch_cos_sim(eng_intend_embedding, elem)
                        for elem in onto_embeddings_list
                  ]
    # Sort descending, along cosine-similarity-values:
    sorted_list_of_embeddings_and_raw_triples = sort(...)
    # Perform PCA to reduce dimensionality to 2D for visualization
    # (PCA-based projection of embedding vectors in 2D):
    pca_points = PCA(n_components=2).fit_transform(
                                    embedding_vectors_list)
    # Return pca_points for visualization and
    # top-3-list of best matches (along with confidence metric):
    returns pca_points_for_vis,
            eng_intention_input_string,
            top3_list_of_best_matches,
            top3_list_of_cos_sim_values
```

Fig. 7. Pseudo code of the NL2IBE core.

The resulting similarity values now allow for sorting the ontological formulation embeddings according to their highest/best suitability. We return the top 3 (or N) closest ontological formulation embeddings along with their similarity values as a measure for confidence and suitability in order to then allow the user to select the best suitable option in the UI/frontend.

Additionally, as a convenience feature, we also provide the user with a 2D-projected visualization of the embedding space and the embedding vectors (of both current engineering intention input and ontological representations), which is obtained by means of running a Principal Component Analysis on the embedding vectors and thereon-based dimensionality reduction. Like this, we can even visualize the proximity of the respective best ontological representations to the given natural language engineering intention input, and thus enable the user to hover around also over other possible representations; see also Fig. 9.

As a remark, if the confidence measure is below a certain threshold, then the tool informs the user that no suitable ontological representation was found, so that – if wanted – a request for appropriate extension of the P&AE ontology can be raised to an ontology engineer.

Summarizing and putting this into context, in this section, we have described how, using our NL2IBE Tool, we are able to transform unstructured/uncontrolled natural language engineering intention inputs into structured/controlled ontological engineering artefacts, which are consistent with the underlying P&AE ontology, and hence suitable for further processing using the tool chain described in Fig. 1. We motivated how one can therefore make intelligent use of NLP methods in combination with knowledge representation and ontologies, which control the final outputs.

Following the engineering intentions processing through our NL2IBE Tool, the collection of ontological triples (or triple combinations) and its combination into an ontologically consistent knowledge graph (with respective instantiations), can be fed into the IBE Tool [4], which in turn is followed by the Pipeline Generation [5] and the Orchestration Designer [6]. This way, once processed through the NL2IBE Tool, we allow for deterministic further processing using the traditional research and product portfolio of ABB's P&AE businesses.

## IV. Evaluation

To illustrate the potential, applicability, and effectiveness of the proposed NL2IBE Tool, we here exemplarily look at a use case scenario from the energy industries, dealing with Hydrogen Generation combined with Carbon Capture Storage and Utilization (CCSU) [21]. An exemplary user wants to build a hydrogen generation and processing plant, see Fig. 8 for the most relevant material transformation processes.

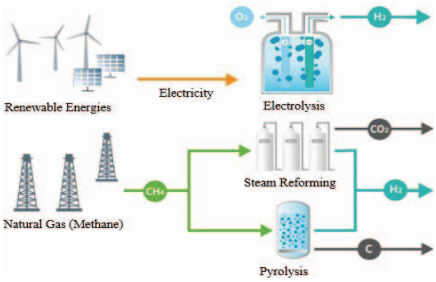


Fig. 8. High-level schema of the material transformation processes for hydrogen generation and CCSU, adapted from [21].

As shown in Fig. 8, processes like electrolysis, steam reforming, or pyrolysis facilitate hydrogen generation (blue/green/grey), depending on the available educts (water or methane), and depending on the type of energy used, as well as further processing, utilization, or storage. Besides the actual processes, along with their input educts and output products, in order for setting up the requirements and process definitions of a future process plant, the user will also want to define the respectively needed plant components or modules, and KPIs or parameters to measure and calibrate the process plant.

Hence, to demonstrate the applicability of the NL2IBE Tool, we assumed and showed that the user can enter all these wanted engineering intentions in natural language and thereon based obtain suitable, consistent ontological representations for further processing in the above-mentioned tool chain. For a screenshot of the UI of the NL2IBE Tool, see Fig. 9.

In the exemplary use case, we investigated and analyzed that the user was able to enter miscellaneous variations of natural language engineering intentions (which describe the above hydrogen generation process) into the given text field, see Fig. 9, and the NL2IBE-Tool would be able to transform these into reasonable and suitable corresponding ontological representations. A top 3 selection of these formalized ontological engineering intention representations is then presented to the user, so that the user can select amongst them the best suitable one, or, in case no representation is satisfying, either try to reformulate the engineering intention in different wording or raise a request to the ontology engineer for appropriate ontology extension, see Fig. 9. Once accepted, the respectively processed transformations are one-after-the-other collected and listed in the session history table, see Fig. 9. This way, we make sure that the thus accumulating ontological formulation equivalents to the human-entered engineering intentions are all well-aligned with the actual intentions, and in their entirety correspond to the overall intended process description. Once done with entering all engineering intentions, the user clicks on 'finalize' and the collection of ontological representations will be transferred into an ontology-consistent instantiated knowledge graph for further processing using the IBE Tool. See Fig. 10, for a screenshot of a such instantiated ontology-based knowledge graph. With this, we have achieved our goal, of providing an easier entry point to the overall P&AE tool chain as shown in Fig. 1.

We remark again that the underlying assumption hereby is that the SemFIL P&AE ontology is capable of representing the intended engineering artefacts: This means, that the respective engineering domain is appropriately represented by means of concepts and interrelations/properties in the ontology *beforehand*, so that the NL2IBE-based transformation and mapping can be executed against it.


Fig. 9. Screenshot of the Frontend/UI of the NL2IBE Tool Prototype.

Of course, we have analyzed and tested our tool also with other relevant use cases, e.g., with a 3-phase oil/gas/water separator, which was thoroughly investigated upon in the context of research on Intent-based Engineering [4,22] and hence as part of the IBE Tool, which is first in the chain of subsequently utilized tools.

Besides our own internally used ontologies, *SemFIL P&AE* ontology [7] and *IBE* ontologies [4], we also tested our NL2IBE Tool against the open *OntoCAPE* ontology [23].

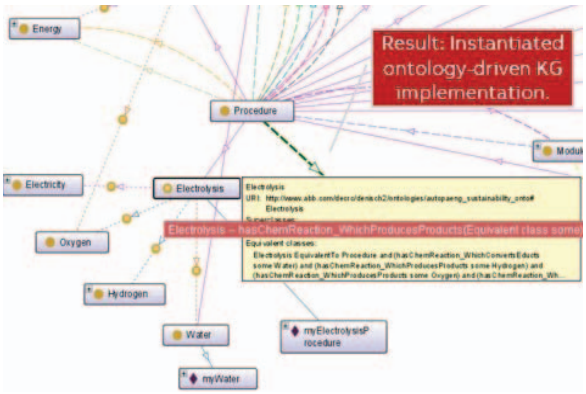


Fig. 10. Screenshot of an exemplary result of the NL2IBE Tool: An instantiated ontology-based KG.

Our tool was well able to translate natural language inputs that resemble the content and domain of *OntoCAPE* ontology into *OntoCAPE*-consistent and -contained ontological representations (i.e., concepts, relations, and triples). We thus see that our tool also here shows suitable understanding and formalization capabilities, however, we have figured out that in this case it is not suitable for our purposes in the context of formalization of engineering intentions. This, however, is not a problem our NL2IBE Tool, but a matter of suitability of the underlying ontology. In our case, *OntoCAPE* is too abstract and too far off the actual content which we aim to represent in this early stage of the P&AE tool chain.

Beyond the above qualitative exemplary considerations, which were investigated upon to prove the subjective suitability of the chosen approach for the given problem, we also quantitatively analyzed our solution method:

On the one hand, a reasonable indication of the adequacy and appropriateness of the proposed method can be obtained by looking at the cosine similarity metric of the respectively obtained top N ontological representations with respect to a respectively entered natural language engineering intention. Here, across all natural language engineering intentions that were entered as part of our evaluation, we obtained an average cosine similarity value of 95.43 for the identified Top 1 results, and of 89.23 for the identified Top 1-3 results, which shows to be a reasonably good and trustworthy result for further processing using the traditional tools mentioned above.

On the other hand, the UI/Frontend of the NL2IBE-Tool allows for a statistical assessment of the average satisfaction of a human expert with the results provided from the tool vs. what the expert had previously entered: As indicated above, upon entering a natural language engineering intention, the NL2IBE-Tool will yield and show the Top 3 most-suitable corresponding formalized ontological representations, so that the user can select among these the respectively best suitable one (or in case no representation is satisfying, either try to reformulate the engineering intention in different wording or raise a request to the ontology engineer for appropriate ontology extension). Our analysis of the human expert satisfaction is shown in Fig. 11.

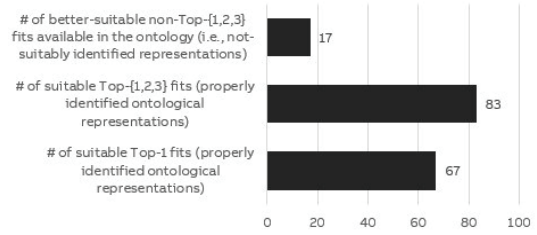


Fig. 11. Assessment of the human expert satisfaction with the results from our NL2IBE Tool across the evaluated intention transformations.

Having evaluated the transformation results from n=100 entered engineering intentions and their corresponding formalized ontological representations, the human expert was best satisfied in 67 cases with the Top-1 proposed ontological representation, and in 83 cases best satisfied with one of the Top-{1,2,3} proposed ontological representations. Opposed to this, according to the human assessor, in only 17 cases there would have been a better-suitable non-Top-{1,2,3} representation available in the ontology, which was not properly identified by the NL2IBE Tool. Overall, this indicates that this tool can simplify the process of entering engineering intentions in roughly 83% of all cases, where the human expert upon entering an engineering intention is directly provided by the tool with a suitable formalized ontological representation, without having to manually construct it by searching and selecting in a tedious process the appropriate concepts, relations, or instances from an ontology or knowledge graph.

We remark that in the evaluation we were not counting the number of requests for ontology extensions, since these would rather allow for a statement towards the suitability and comprehensiveness of the underlying ontology, as opposed to giving insights for the evaluation of the NL2IBE Tool itself.

## V. Summary, Discussion & Outlook

### A. Summary & Discussion

Summarizing, in this work, the authors present the NL2IBE Tool. The purpose of this tool is to support and to substantially simplify the very early engineering phase of entering engineering intentions into a set of existing engineering tools, see again Fig. 1. To date, these intentions either needed to be entered in an already structured and controlled formal expert language, or they required a human expert's subsequent manual efforts to translate unstructured and uncontrolled natural language into formalized (controlled and structured) representations. Only upon formalization they could then be used for *consistent and deterministic* further processing using the traditional engineering tools.

So far, the established process, which includes having a human expert's efforts to help with the translation and formalization, is very time-consuming, fuzzy, and also error-prone due to frequent human misconceptions and semantic ambiguities. On the contrary, with the proposed NL2IBE Tool, we strongly support and simplify this process in roughly 83% of all cases, leaving only roughly 17% of cases for manual formalization efforts as before. Without requiring the user to bother about formal and structural correctness of inputs, our tool proposes and converts uncontrolled text inputs into structured ontological representations, which are *inherently consistent* with an underlying ontology. Our NL2IBE Tool therefore makes use of modern NLP methods combined with the intelligent extraction of ontological structures in information models and domain knowledge representations.

Particularly, we achieve formalization of uncontrolled inputs by transforming them into their abstract/latent embedding representations in a *joint* embedding space with controlled and structured ontological formulations (i.e., concepts, relations, and triples, as well as triple combinations from the P&AE ontology), followed by subsequent finding of respectively closest ontological representation equivalents.

With this setup, we enhanced the previously purely transformers-based solution, which itself is not directly controllable, not transparent, not guaranteed to be reliable, and not guaranteed to be consistent with underlying ontological information models either, by means of a combination with an ontology-based controllable, consistent, reliable & transparent mapping from transformed engineering intention representations to their respective ontological representation equivalents.

NL2IBE thus not only translates and formalizes natural language intentions, but also, based on and controlled by the underlying P&AE ontology, takes out the *variation* and

*randomness* of natural language, by mapping uncontrolled inputs to a finite set of structured and controlled ontology-consistent engineering intention artefacts as outputs. We consider this *robust-making* and *de-randomization* crucial, particularly for critical processes, systems, and critical infrastructure, where purely-transformers-based probabilistic 'black box' solutions are not acceptable, neither by customers nor by the regulation institutions.

With the structured and ontology-consistent output generated by the NL2IBE Tool, we have achieved the goal, of providing an easy, flexible, and robust entry point to the overall P&AE tool chain as shown in Fig. 1. Once processed through the NL2IBE Tool, we allow for deterministic further processing using the traditional research and product portfolio of ABB P&AE.

### B. Outlook & Future Work

As part of future work, we envision several points of improvement, extension, and generalization:

Firstly, we plan to have an ontology-based and ontology-driven extension of the NL2IBE Tool, which helps and pushes the engineer to *further extract more* engineering intention details and to further *augment* or *complete* the so-far-entered formalized ontological engineering intention representations. This means that, when an engineering intention was entered in the above described way using our tool, the system will check and propose to the human engineer what ontologically related features may subsequently still need to be entered and defined next. It may thus trigger missing pieces of information, ask to give more details in incomplete ontological representations, and even propose entirely augmented setups, by comparing the current state of representation in a current situation with historical similar situations.

Second, we have in mind several UI and UX improvements for *easier and more intuitive usability*, and we will want to further analyze these also based on user tests.

Third, we shall make use of the expert feedback which is provided to us through the UI of the NL2IBE Tool, when the user selects among the top 3 ontological representations the respectively best suitable one. This information can be fed back to the transformer-NN-based part of the NL2IBE algorithm and have an impact on the weights and biases for the *embedding computation*.

Last but not least, we are working on the generalization of the NL2IBE core idea towards applying it not only to engineering intentions (and their formalization), but to *transferring the core functionality for utilization also with other input/output types and schema*, e.g., P&AE control narratives and other unstructured input data, and as output not only ontological representations, but also other structured formats, such as RDF, XML, AML, JSON, etc.

## REFERENCES


[1] M. Hoernicke, T. Knohl, J. Bernshausen, H. Bloch, A. Hahn, S. Hensel, A. Haller, A. Fay, L. Urbas: „**Steuerungsengineering für Prozessmodule – Standardkonforme Modulbeschreibungen automatisch erstellen**". atp edition 59(3): 18-29, 2017.

[2] M. Hoernicke, K. Stark, T. Knohl, J. Bernshausen, H. Bloch, A. Fay, A. Menschner, S. Hensel, L. Urbas, A. Haller, G. Lustig: "**Orchestration of Modular Plants"**. Achema 2018, Frankfurt, 2018.

[3] M. Hoernicke, K. Stark, N. Schoch, R. Jeske, A. Markaj, A. Fay: "**Modular Engineering of Conventional Plants - Using MTP for World-Scale Industry Plants**". atp edition 64(4):31-47, 2022.

[4] A. Markaj, N. Schoch, K. Stark M. Hoernicke, A. Fay: "**Intention-based engineering for process plants**". IEEE SysCon 2022.

[5] N. Schoch, M. Hoernicke, K. Stark: "**Semantic function module pipeline generation**". Journal of at - Automatisierungstechnik, 2021.

[6] M. Hoernicke, C. Messinger, E. Arroyo, A. Fay: "**Topology Models in AutomationML – Object-oriented basis for the automation of automation**". atp edition 58(5): 28-41, 2016.

[7] N. Schoch, M. Hoernicke, K. Stark: "**Semantic Facilitation and Integration Layer for Process and Automation Engineering**". IEEE ETFA 2023.

[8] T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, et al.: "**Language Models are Few-Shot Learners**". NeurIPS-33, 2020.

[9] W.X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, et al.: "**A Survey of Large Language Models**". ArXiv:2303.18223, 2023.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, et al.: "**Attention is All you Need**". NeurIPS-30, 2017.

[11] J. Kaddour, J. Harris, M. Mozes, H. Bradley, R. Raileanu, R. McHardy: "**Challenges and Applications of Large Language Models**". ArXiv:2307.10169, 2023.

[12] P. Schneider, T. Schopf, J. Vladika, M. Galkin, E. Simperl, F. Matthes: "**A Decade of Knowledge Graphs in Natural Language Processing: A Survey**", ArXiv:2210.00105, 2022.

[13] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, X. Wu: "**Unifying Large Language Models and Knowledge Graphs: A Roadmap**", ArXiv:2306.08302, 2023.

[14] A.S. Behr, M. Völkenrath, N. Kockmann: "**Ontology extension with NLP-based concept extraction for domain experts in catalytic sciences**". J. Knowledge & Information Systems, 2023.

[15] J. Sequeda, D. Allemang, B. Jacob: "**A Benchmark to Understand the Role of Knowledge Graphs on Large Language Model's Accuracy for Question Answering on Enterprise SQL Databases**". ArXiv:2311.07509, 2023.

[16] Q. Wang, Z. Mao, B. Wang, L. Guo: "**Knowledge graph embedding: A survey of approaches and applications**". IEEE Transactions on Knowledge and Data Engineering 29(12), 2017.

[17] J. Chen, P. Hu, E. Jimenez-Ruiz, O.M. Holter, D. Antonyrajah, I. Horrocks: "**OWL2Vec - Embedding of OWL Ontologies**". ArXiv2009.14654v2 2021.

[18] J.B. Lamy: "**Owlready - Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies**". Artificial Intelligence In Medicine, 80:11-28, 2017.

[19] N. Reimers, I. Gurevych: "**Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks**". Conf. on Empirical Methods in NLP, 2019.

[20] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, et al.: "**HuggingFace's Transformers: State-of-the-art NLP**". Arxiv:1910.03771, 2019.

[21] BDEW: "**Hydrogen – Wasserstoff – Small molecule with huge potential**". www.bd.ew.de (BDEW), 2021.

[22] A. Markaj, A. Fay, N. Schoch, K. Stark, M. Hoernicke**:** "**Intention-based engineering for the early design phases and the automation of modular process plants**". ETFA, 2022.

[23] W. Marquardt, J. Morbach, A. Wiesner, A. Yang: "**OntoCAPE - A Re-Usable Ontology for Chemical Process Engineering**". RWTHedition (Book), 2012.