FLYING FORWARD 2020

Natural Language Understanding for Drone Legislation

DATE:
27-09-2023

AUTHORS:
Peter Bourgonje
Geert Devos
Jan Van Sas

Flying Forward 2020 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no.101006828.

Table of Contents

1 Introduction

This white paper presents Nalantis' solution to extracting structured, queryable information from unstructured, natural language texts containing legislation on Urban Air Mobility. This approach has been developed in the context of the Flying Forward 2020 project[1], a collaborative three-year research and innovation project developing state-of-the-art Urban Air Mobility (UAM).

In the Flying Forward project, we collaborated with several industry and research partners, each working on their own area of expertise. As a company specialized in Natural Language Processing (NLP), Nalantis was responsible for developing the framework to make sense of unstructured, plain legal text. While legal texts might seem highly formal, structured and rigid from a human perspective, they are still a long way from allowing accurate and reliable machine-based parsing. Despite the apparent formality, these texts, exhibit a lot of ambiguity and the use of under-specified constructions is common. Several steps need to be performed to convert natural language sentences into a structure that machines can act upon. In this white paper, we explain these steps and how we implemented them programmatically.

[1] https://www.ff2020.eu/

Automated processing of natural language (often referred to as NLP, or NLU, for Natural Language Understanding) and its related research field have undergone dramatic changes in the last decade, aligning with recent developments in AI (Artificial Intelligence), of which NLP can be considered a sub-discipline. Increasing computational power and the availability of large data sets enabled vectorized representations of natural language, which were static at first, but contextualized later. Static embeddings (*Mikolov et al., 2013*) represent the same word with the same vector every time, regardless of what context it occurs in. Contextualized embeddings (*Devlin et al., 2018, Peters et al., 2018, Yang et al., 2019*) take the context of a word into account, and the occurrence of "*context*" in this very sentence, preceded by the word "*the*" and succeeded by the word "*of*", would be represented by a different vector than the one for "*context*" in the previous sentence, which is preceded by "*what*" and succeeded by "*it*". The current generation of such semantic representations of natural language, using ever-more data to train on, and more sophisticated (neural) models to represent input, are Large Language Models (LLMs). Since the introduction of LLMs, most notably ChatGPT[2], many NLP applications (question answering, summarisation, machine translation, etc.) are approached in an end-to-end way, meaning that the same model is used to perform an entire task from start to end. The user asks the model to perform a task as they would ask a human, and obtains an answer, ideally as a (knowledgeable) human would answer. While progress using the underlying pretrained transformer models has been impressive, a significant part of the interpretation and evaluation load remains on the human processing the LLM answer. Both in terms of format (specifically instructing, for example GPT-4, to output JSON in a specific format does not always produce the required results), and in terms of correctness and reliability. Our solution, using a no-black-box approach, aims to address these shortcomings, by producing explainable output from the AI system, which decreases errors and increases an end user's confidence in the results returned.

In particular, we argue that such an approach to UAM legislation processing should be **accurate**, in the sense that it targets the relevant sentences and extracts information from them (usually, sentences expressing some deontic type: shall, should, must, etc.), and ignores sentences that contain no actionable information. E.g

> Sentence containing deontic information that should not be skipped:"*The unmanned aircraft shall be maintained within 120 metres from the closest point of the surface of the earth.*"
>
> Sentence with no actionable information: "*Subject to paragraph (4), there is one runway protection zone for each runway threshold of each runway at the aerodrome.*".

The approach is also highly **reliable**, in that it delivers consistent results, and can be trusted to provide the same answer when asked the same question, also when

[2] https://chat.openai.com

there is some time in between asking questions. Despite remarkable progress in NLP with generative pretrained models (Large Language Models), this is not a given with many LLM approaches (*Chen et al., 2023*).

Finally, the approach is easily **customizable**, by which we mean that we have control over the output, and can correct things that are going wrong, which is also not a given when using LLMs (prompt engineering influences output structure and quality, but its potential to control the output is comparably limited).

1.1 Targeted Users

Our solution is targeted at both drone operators and drone manufacturers. The former will want to know, either before or during a flight, how to plan their route in advance or change their status in-situ in order to comply with the relevant laws. The latter will want to know what their products have to look like to comply with local legislation, or conversely, in which geographical areas (countries, states, cities, etc.) their drones are allowed to fly. Because different governmental bodies (nations, cities, rural areas, etc.) and U-Space providers[3] each tend to have their own rules (with certain basic principles shared among nations or even continents), an automatic procedure to convert natural language legal texts into machine-readable, unambiguous and queryable knowledge, can drastically speed up the process of planning or controlling drone flights (for operators), or getting a product to market (for manufacturers).

In the context of the Flying Forward 2020 project, there have been several so-called Living Labs, showcasing the innovation in the project. For the drone flights performed during these showcases, the operators had to plan their route prior to the event and supervise the flight itself, all the time being aware of the relevant rules and regulations themselves. This involves being aware of the dimensions (size, weight, etc.) of the drone itself, the payload of the drone, the vicinity of no-fly areas, crowds of people spontaneously forming on the ground, the minimum flying height of the drone and the height of buildings in the area, and many, many more factors relevant for the drone's flight. We aim to (semi-)automate this process. Assuming that the state the drone is in is known to the drone and/or its operator, we parse legal texts into machine-readable structures such that compliance with local regulation can be checked at any time. Ideally, we can also speed up the planning and pre-flight preparation processes.

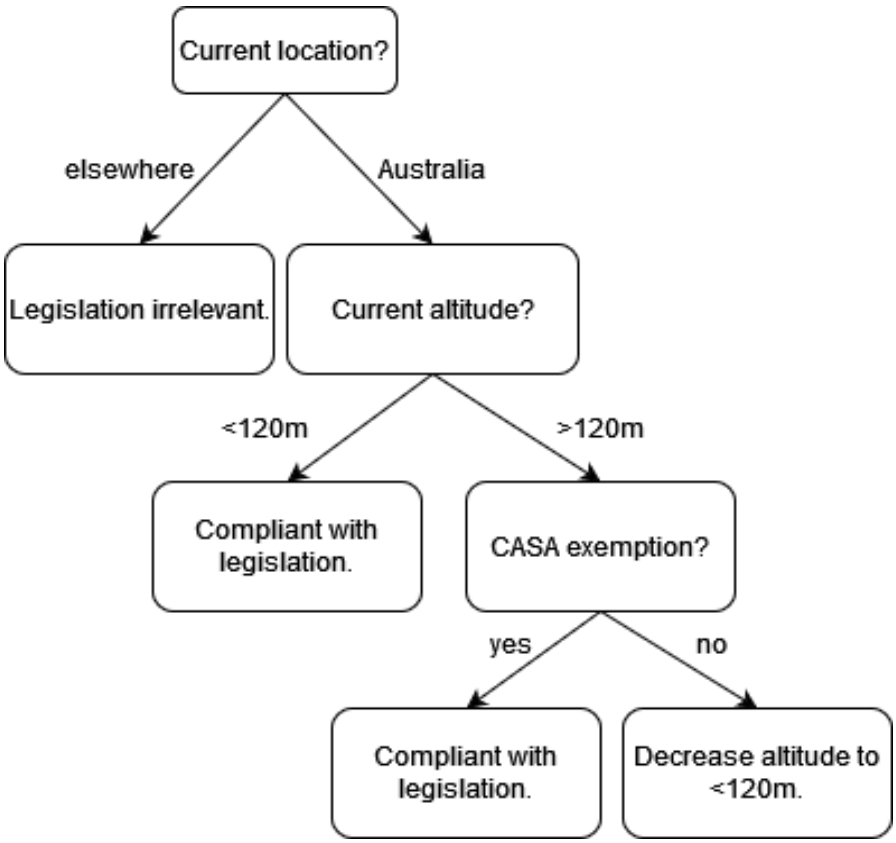
1.2 Text to Structure

Our solution operates on plain, natural language sentences, and converts this to structured, machine-readable knowledge. While the specific output format (and thus, representation of knowledge) is a matter of research itself, to provide one example of how we aim to convert natural language into a structure that enables

[3] https://www.ecac-ceac.org/activities/unmanned-aircraft-systems/uas-bulletin/22-uas-bulletin/504-uas-bulletin-2-what-is-u-space

drones or pilots to directly act upon regarding compliance, consider the following sentence:

"*In Australia, drones cannot be flown higher than 120 meters (approximately 394 feet) above ground level, unless an exemption is obtained from the Civil Aviation Safety Authority.*"



Assuming that the current state of a drone is something that is known by an operating system or operator controlling the drone, checking this against such a structured, decision-tree like representation of legislation allows the pilot or operating system itself to be compliant with local or (inter)national rules, and adjust its situation if necessary.

For drone manufacturers, such inferences can automatically provide them information about which markets their products can be offered. While the above sentence expresses a maximum flight altitude, legislation might for example also include sentences stating a minimum altitude, and drones that are unable to reach that altitude will thereby disqualify them for those areas and the corresponding regions or countries. This requires parsing natural language legislative texts into machine-readable structures, and adding inference to that structured representation.

The conversion of plain text to actionable, decision-tree like schemas is easily done by humans, but extracting them from plain text is non-trivial. In fact, it is easy to overlook exactly how effective humans are at inferring information from text. Consider the following sentence: "*Pilots must maintain a safe distance from other aircraft and avoid collisions.*"

Even though the sentence has the pilot as its grammatical subject, human readers will easily infer that it is not just the pilot, but perhaps even more so the drone the pilot is piloting, that has to maintain a safe distance with other aircraft to avoid a collision, despite there being no explicit mention of a drone (that the pilot is commandeering) in the sentence.

Many of the relevant sentences in legislation (i.e., sentences containing some kind of actionable information) can be classified according to their deontic type (*von Wright, 1951*). Consider the following sentence: "*The minimum age for remote pilots operating a UAS in the 'open' and 'specific' category shall be 16 years.*"
While this is very straightforward, and expresses a condition that simply must be met (i.e., the pilot must be 16 years or over), legal sentences are not always that clear. Consider the following: "*The UAS Operator aims to carry out UAS operations in a way that minimizes the risk of damage to property or injury to persons.*", where the modal verb "*aims*" expresses an attempt to meet the regulation, but is less clear on the consequences if this condition is not met.

The extraction of deontic types is crucial to interpreting legal texts (the reader will want to know what is and is not allowed, under which conditions, which are the exceptions, etc.). These can often be inferred from modal verbs, potentially negated (shall, will not, must, cannot, etc.), but can also be expressed in different ways, for example, through an imperative ("*Take witness statements if appropriate.*").

The above examples serve to demonstrate that the task we set out to do is non-trivial. An accurate, reliable and customizable solution is needed to convert natural language into structured information (JSON, XML, YAML or similar) to make its content accessible to machine-based reasoning.

1.3 Extendability

While the approach outlined in this white paper is developed with UAM legislation in mind, the core engine can be used in other contexts as well. For the Flying Forward 2020 project, we have been populating linguistic resources to ensure high performance on UAM legislation, but the NLP engine exploiting these domain-specific, linguistic resources, is domain-independent and multilingual.The core linguistic analysis is using a Python library (spaCy) that supports a wide range of languages, and the layer that was developed on top of that uses rules and mechanisms that are applicable to other (European) languages and domains straight away. Adaptation to a new domain or language requires the creation of a specific ontology for that domain or language. In the context of the Flying Forward 2020 project, we have been doing this manually, but for follow-up use cases, several steps can be automated to speed up this process.

2 Legal SAGE

Within the domain of Natural Language Processing (NLP), there are several popular use cases that approximately address the problem we aim to solve. Parts of the use case described in the sections above are tackled by Question Answering (Wang, 2022), others by Text Summarization (Awasthi et al., 2017), and natural language SQL querying, for which there already exists a vast amount of work (including, Zhong et al., 2018, Guo et al., 2019, Yu et al., 2018). In recent years, more of a one-size-fits-all approach to several NLP use cases has rapidly gained popularity, primarily using LLMs.. While solving the problem outlined in Section 1 quite accurately in some cases, we see two major challenges with a solution being (solely) based on LLMs:

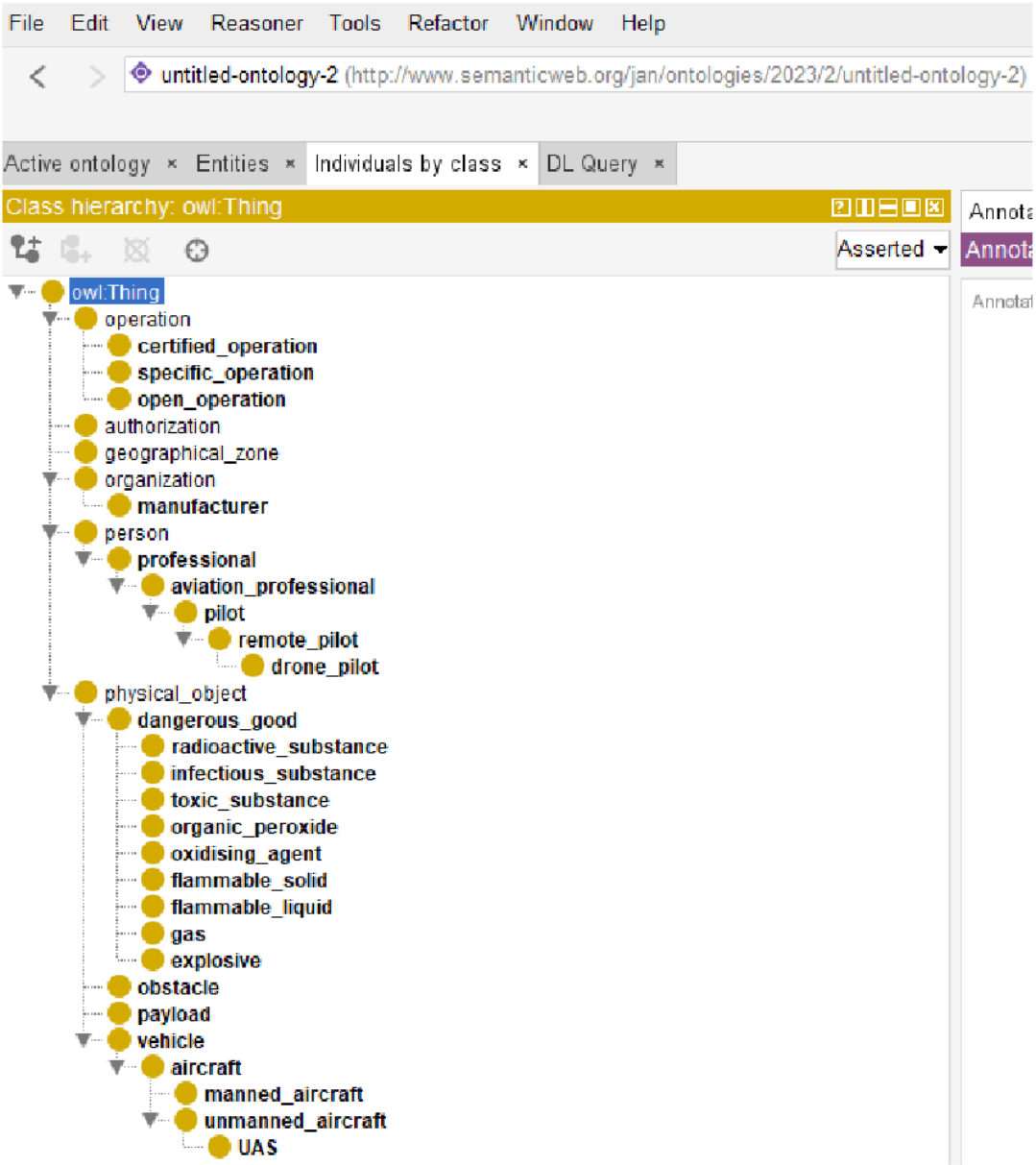
1. **Reliability**. Prompt engineering and hyperparameter setting and tuning might significantly decrease the amount of "hallucination" (a known problem with LLMs), and output can be required to be in some machine-readable format (JSON, for example). However, ensuring that the given answer to a question is accurate, reliable and machine-readable remains highly challenging. While this may be true for any automated system (i.e., also for entirely rule-based systems, output quality is never likely to be 100% correct), this leads to the second challenge.

2. **Customizability**. We strive for a no-black-box solution, where output can be changed if it does not live up to quality standards, if specific matches or answers are undesired, or if desired matches do not show up in the output. Building upon more modular NLP pipeline components enables us to have more control over the output, both in terms of format and content.

The following subsections explain some of the key modules in the Nalantis Legal SAGE (Semantic Analysis Generation Engine) pipeline, going from plain text input to machine-readable information extracted from that input.

2.1 Concept Matching

After basic sentence segmentation and tokenization (the first, very basic step in NLP pipelines, that recognizes sentences and splits them up into their individual words), we link single words and multi-word phrases to concepts in our ontology. The ontology organizes domain-relevant concepts in a hierarchy that is instrumental to allowing reasoning and going from surface form to understanding. For example, we include “unmanned aircraft” as a lemma (word-based matching is based on lemmatization), of the class *UnmannedAircraft*, which is a subclass of *Aircraft*, which in turn is a subclass of *Vehicle*, *PhysicalObject* and finally *Thing*. This hierarchical setup enables inheritance, and we can apply properties of superclasses to subclasses. First, this allows for highly domain-specific matching and interpretation of words and phrases related to drone legislation. Second, real-world knowledge of concept properties can be included and used as-is, or for the validation of downstream processes. An excerpt of our drone ontology is included in the following illustration:



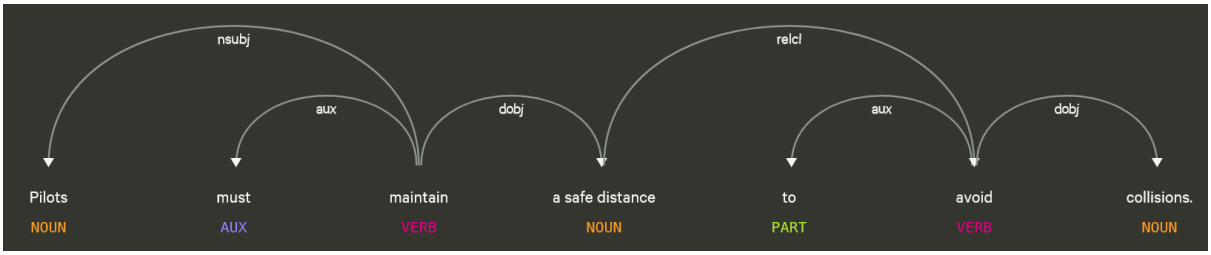
Population of the entire ontology was aided by several existing drone/law ontologies[4], but the ontology used for the Flying Forward 2020 project is unique, domain-specific, Nalantis intellectual property that comprises the in-house backbone of our approach for automated drone legislation processing.

[4] https://op.europa.eu/en/web/eu-vocabularies/eli , http://www.dronetology.net/dronetology/index-en.html , https://data.nasa.gov/ontologies/atmonto/

2.2 Dependency Parsing

After matching concepts at token level (single words or multi-word units), we exploit the dependency parser from spaCy[5] to extract a dependency graph from an input sentence. This procedure is basically a normalization step to arrive at a semantic representation that encodes the roles of various constituents in the sentence, and is able to abstract away from low-level surface form differences.

A simple example is included in the graph below:



Where we can infer several things from the graph structure:

- The key message of the sentence (i.e. that pilots must maintain a safe distance) can be extracted by looking at the so-called root node of this graph (i.e., the node with no outgoing edges and only incoming edges, the verb "maintain" in this example).
- This key message can be expressed in an RDF-like triple, encoding predicate and arguments, in this case something like *maintain(pilot, safe distance)*.
- Exceptions, specifications or conditions can often be inferred by looking at clauses that are connected to the root node (or any subclause) through, for example, a relcl-type relation (for "relative clause").

The advantage of working with dependency graphs, is that it is more robust than working with surface forms directly, and it abstracts over singular/plural differences, morphological and other variations that essentially express the same thing semantically, and generally aims to represent the function of words and phrases in context.

The spacy library comes with a special matcher[6] that allows the formulation of a set of rules that specify what information to extract from which dependency graphs. Generally, we focus on the core relation triple, expressing the verb and its subject and (in the case of a transitive verb) direct object, and any conditions or requirements linked to this main relation triple.

[5] https://spacy.io/

[6] https://spacy.io/api/dependencymatcher

2.3 Thematic Roles, Deontic Logic & Verbnet Validation

After extracting RDF-like triples from the dependency graph, we return to our ontology to validate these triples. Consider the following example sentence:

“The UA should not carry dangerous goods, except for dropping items in connection with agricultural, horticultural or forestry activities.”

From the dependency graph, we extract that the main verb of this sentence is “carry”, which has “The UA” as its subject, and “dangerous goods” as its direct object. The verb “carry” has the following entry in our ontology:

```
Carry:
  syns:
  - carry
  - transport
  - bring
  super: Frame
  syntax:
  - Agent ActionVerb Theme
```

Expressing that this entry has some synonyms (“transport” and “bring”) conveying the same meaning, is a subclass of the Frame class, and the “syntax” information links syntax with thematic roles (Chomsky, 1981, Carlson, 1984) by encoding that the subject of this verb is an Agent, and the direct object a Theme. If we now consider the ontology entry for “UnmannedAircract” and its superclasses:

```
UnmannedAircraft:
  syns:
  - drone
  - UA
  - UAS
  - UAV
  - unmanned aircraft
  super: Aircraft
  props:
  - isManned(False)
```

```
Aircraft:
  syns:
  - aircraft
  - airplane
  super: Vehicle
  props:
  - MTOM
  - TakeOffWeight
```

```
Vehicle:
  syns:
  - vehicle
  super: PhysicalObject
  props:
  - Payload
  - Loudness
  - Velocity
  - Pilot
  - Agent
```

We can see that “UnmannedAircraft” inherits the “Agent” property from its (indirect) superclass “Vehicle”, meaning that we can use the ontology to validate the semantic frame encoded in the ontology entry for the verb “carry”.
Thus, combining the dependency parse and the ontology information, we arrive at the following interpretation for the core triple in the example sentence:

```
Carry_(18-23):
- type: Prohibition
  pred: Carry
  Agent:
  - UnmannedAircraft
  Theme:
  - DangerousGood
```

The above is a human-readable, indented version of the internal representation format (JSON), which is fully machine-readable.
The verb (“carry”) along with its character offsets (since there might be more than one occurrence of this token in the sentence, we mark it so it remains uniquely identifiable) is validated with the Agent “The UA” and the Theme “dangerous goods”.
Furthermore, we can infer the deontic type of this sentence from the modal verb “should” and the negation “not” that has scope over this modal verb, resulting in the deontic type of *prohibition*. This information is equally coming from the ontology, where the entry of “should” is enriched with the type “Obligation”, which, when negated, is converted to “Prohibition” in our analysis.

Our ontology is the first go-to for enriching and validating information coming from the upstream process. If we have not found any information regarding a specific verb however, we look it up on verbnet (Schuler, 2006), an open-source resource for (verb) frame semantics, from which information about synonyms and thematic roles can be extracted. In our experience, verbnet (and its noun-oriented predecessor wordnet) tends to over-generate, has too wide “synsets” (groups of words which are taken to be synonyms), and encodes rather too much than too little information (i.e., including archaic, domain-specific or colloquial usage of words which are not relevant for our domain). This is why verbnet is used only as a fall-back option in case no information about the specific verb is found in our own ontology.

2.4 Output Examples

The following screenshots illustrate some input text and the structured representation in JSON/YAML format as the final result of the analysis described in the sections above.

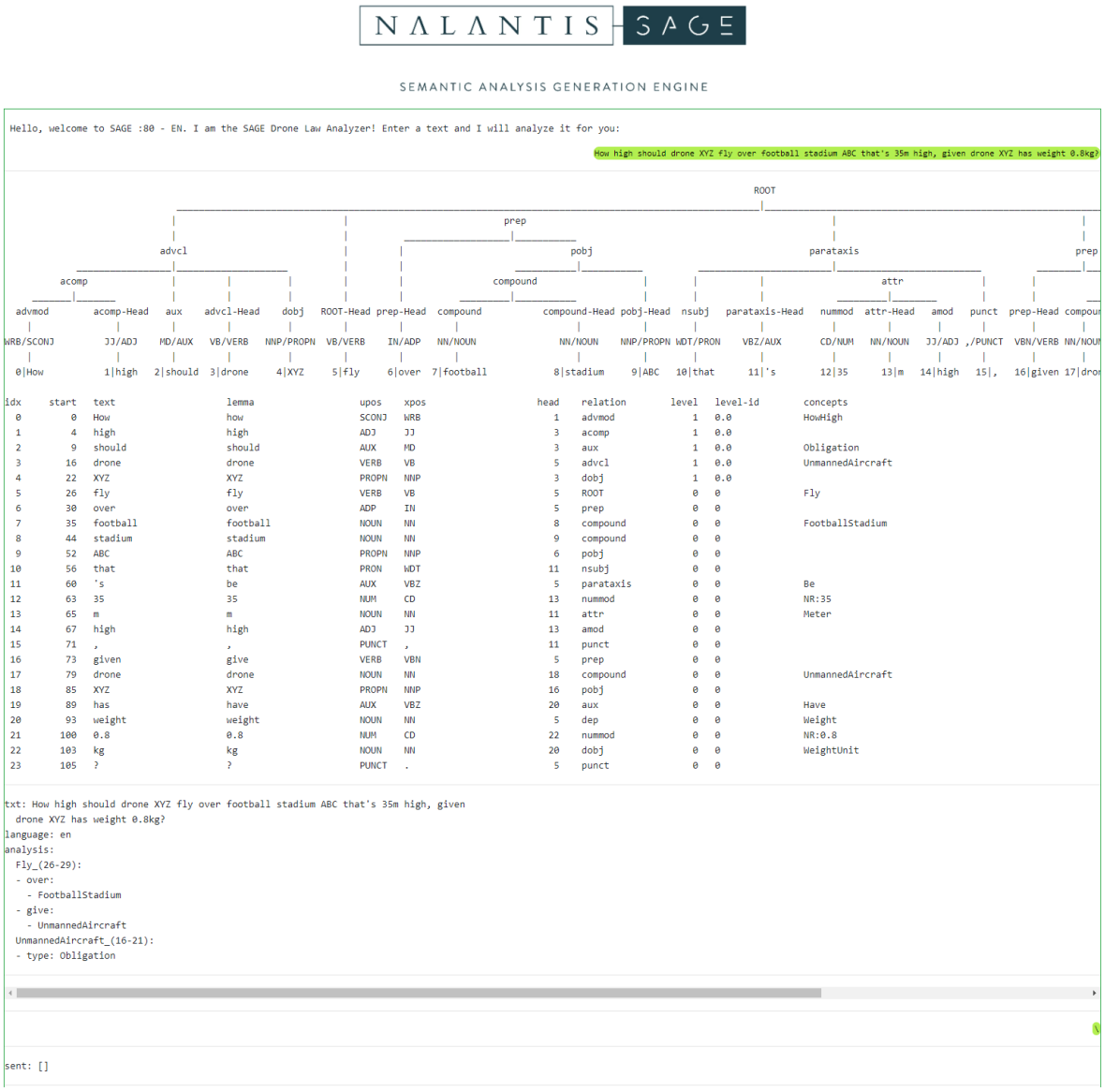

```
{
  "txt": "Drones must not exceed a maximum noise level of 80 decibels when measured at a distance of 50 meters.",
  "language": "en",
  "analysis": {
    "Exceed_(16-22)": [
      {
        "type": "Prohibition",
        "pred": "Exceed",
        "Agent": [
          "UnmannedAircraft"
```

```
            ],
            "obj": [
                "Maximum",
                "Loudness",
                "NR:80",
                "Decibel"
            ]
        },
        {
            "condition": [
                "NR:50",
                "Measure",
                "Distance",
                "DistanceUnit"
            ]
        },
        {
            "condition": [
                "NR:50",
                "Measure",
                "Distance",
                "DistanceUnit"
            ]
        }
    ]
  }
}
```

2.5 Question Answering

The subsections above explain how we arrive at a structured representation of natural language sentences. However useful such a representation is, obtaining this is hardly a goal in itself. It is the downstream applications it enables that are of most interest. For example, we can use it to facilitate question answering.

The small demo outlined below, assumes an empty system that has not processed any legal texts before. If it is confronted with the question “How high should drone XYZ fly over football stadium ABC that’s 35m high, given drone XYZ has a weight 0.8kg?”, a structured representation of the question is generated (just like for any other sentence, from legal texts), but given that the system has not “read” any legal texts yet, no answer is available.



If we now feed the system a legal statement, it is analyzed and its representation is added to the system’s memory.

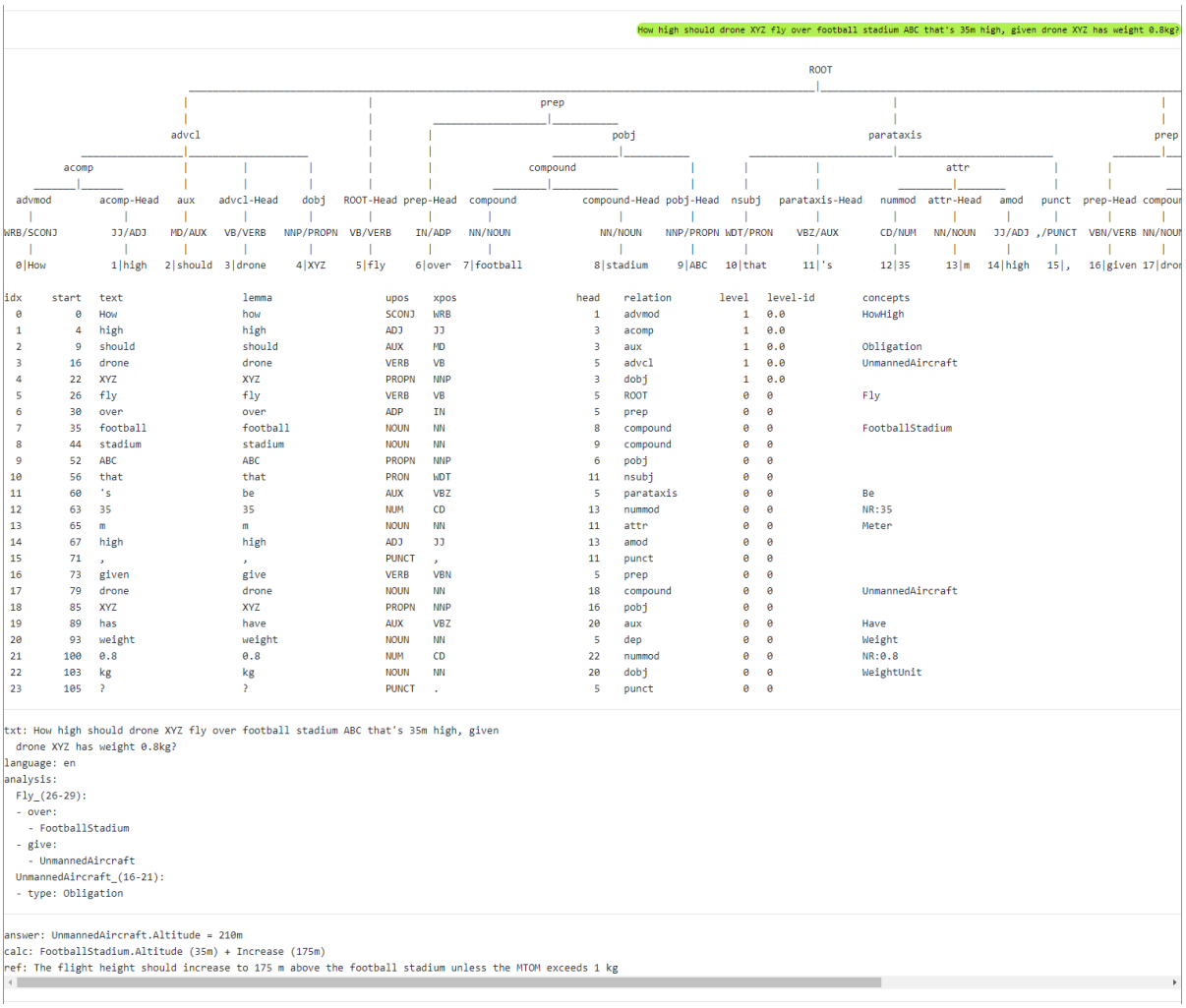
```
The flight height should increase to 175 m above the football stadium unless the MTOM exceeds 1 kg

                                                 ROOT
       ___________________________________________|___________________________________________
      |                |       |       prep                  prep                          advcl
      |                |       |     ____|____         _______|_____              ___________|____________
    nsubj              |       |    |       pobj      |            pobj          |       nsubj     |        dobj
   ___|_____           |       |    |     ___|___     |        _____|_____       |     ___|___     |      ___|___
det  compound nsubj-Head  aux  ROOT-Head prep-Head nummod  pobj-Head prep-Head det  compound pobj-Head  mark  det  nsubj-Head advcl-Head nummod  dobj-Head
 |      |       |       |       |        |       |          |         |       |       |        |        |     |       |          |        |        |
DT/DET NN/NOUN NN/NOUN MD/AUX VB/VERB  IN/ADP  CD/NUM    NN/NOUN   IN/ADP  DT/DET  NN/NOUN  NN/NOUN IN/SCONJ DT/DET NNS/NOUN  VBZ/VERB  CD/NUM  NNS/NOUN
 |      |       |       |       |        |       |          |         |       |       |        |        |     |       |          |        |        |
0|The 1|flight 2|height 3|should 4|increase 5|to 6|175    7|m     8|above  9|the  10|football 11|stadium 12|unless 13|the 14|MTOM 15|exceeds 16|1  17|kg
```

idx	start	text	lemma	upos	xpos	head	relation	level	level-id	concepts
0	0	The	the	DET	DT	2	det	0	0	
1	4	flight	flight	NOUN	NN	2	compound	0	0	Flight,Fly
2	11	height	height	NOUN	NN	4	nsubj	0	0	Altitude
3	18	should	should	AUX	MD	4	aux	0	0	Obligation
4	25	increase	increase	VERB	VB	4	ROOT	0	0	Increase
5	34	to	to	ADP	IN	4	prep	0	0	
6	37	175	175	NUM	CD	7	nummod	0	0	NR:175
7	41	m	m	NOUN	NN	5	pobj	0	0	Meter
8	43	above	above	ADP	IN	4	prep	0	0	GreaterThan
9	49	the	the	DET	DT	11	det	0	0	
10	53	football	football	NOUN	NN	11	compound	0	0	FootballStadium
11	62	stadium	stadium	NOUN	NN	8	pobj	0	0	
12	70	unless	unless	SCONJ	IN	15	mark	1	0.0	Exception
13	77	the	the	DET	DT	14	det	1	0.0	
14	81	MTOM	MTOM	NOUN	NNS	15	nsubj	1	0.0	MTOM
15	86	exceeds	exceed	VERB	VBZ	4	advcl	1	0.0	Exceed
16	94	1	1	NUM	CD	17	nummod	1	0.0	NR:1
17	96	kg	kg	NOUN	NNS	15	dobj	1	0.0	WeightUnit

```
txt: The flight height should increase to 175 m above the football stadium unless
  the MTOM exceeds 1 kg
language: en
analysis:
  Exceed_(86-93):
  - pred: Exceed
    subj:
    - MTOM
    obj:
    - WeightUnit
    - NR:1
  Increase_(25-33):
  - type: Obligation
    pred: Increase
    subj:
    - Flight
    - Altitude
    - Fly
  - exception:
    - Exceed
  - to:
      Meter:
      - NR:175
  - above:
      GreaterThan:
      - FootballStadium
```

```
sent: ['The flight height should increase to 175 m above the football stadium unless the MTOM exceeds 1 kg']
```

If we now ask the system the same question again, it will search its memory, or knowledge base, for relevant answers. For the given question, the relevant answer is generated based on legislative sentences in the knowledge base:



By iteratively feeding it legal sentences (i.e., just entire legislative texts, for example the EU regulation on the rules and procedures for the operation of unmanned aircraft[7]), we can populate a knowledge base, and answer questions relevant to the legislation in question.

3 Challenges and Future Work

Recall from Section 1 that we strive for a system that is accurate, reliable and customizable. While the rule-based system described above has the potential to fulfill these requirements, one of the main drawbacks is the manual labor that goes into carefully constructing the ontology that is the backbone of our solution. Our drone law ontology has been constructed and populated by a team of three linguists over the course of several months. Populating and maintaining the ontology requires no programming knowledge, but it is a labor-intensive process that requires an understanding of how our system works, as well as experience in ontology building and/or population, and familiarity with existing EASA and ISO documents to ensure consistent terminology of classes, synonyms and properties.

[7] https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32019R0947

In addition to further expanding our drone ontology, we plan to experiment with additional domains (other than UAM), and will try out semi-automated approaches. One promising direction of future work will be finding domain-specific equivalents of ontology entries in an automated way. By getting the representation of ontology items in vector space (by training an embeddings model on UAM legislation), we hope to speed up the process of finding relevant items. This procedure could enable us to quickly find the domain-specific equivalent of “drone manufacturer” in another domain. In the healthcare domain for example, a good equivalent could be a construction company (constructing the building housing a hospital). One of the traditional use cases for word embeddings (Mikolov et al., 2013) was to answer questions like “Man is to woman what king is to X.”, where the model was supposed to substitute X for “queen”. The difference, in our case, is that we would have two embeddings models (as opposed to just one in the traditional case), and combining two or more models in a sensible way remains a research question.

Many legislative texts reference other legislations, or specific paragraphs thereof. Our current analysis is aimed at single texts, and references to other legislation should be resolved as a preprocessing step. Doing this in a more sophisticated way represented another direction of future work we intend to do. In this regard, we are particularly interested in developing modules that deal with legislation changing over time, and automatically detecting which relevant bits and pieces of legislation (relevant to some specific product, user or use case) have changed.

4 Conclusion

In this whitepaper, we outlined our solution to processing natural language legal sentences from the domain of Urban Air Mobility to structured, machine-readable representations. A collection of sentences, converted into such representations, will then effectively become a knowledge base, which can be queried, using questions formulated in natural language. While recent developments in NLP, particularly in the area of Large Language Models, have seen impressive performance on a number of tasks and use cases, we argue that these approaches suffer from lack of accuracy for specific, non-general domains, and are limited in their reliability and customizability. Our largely rule-based approach aims to address these shortcomings.

References

Ishitva Awasthi, Kuntal Gupta, Prabjot Singh Bhogal, Sahejpreet Singh Anand, Piyush Kumar Soni, 2017. *Natural Language Processing (NLP) based Text Summarization - A Survey.*

Greg Carlson. 1984. *Thematic roles and their role in semantic interpretation*.

Lingjiao Chen, Matei Zaharia, James Zou, 2023. *How is ChatGPT's behavior changing over time?* https://arxiv.org/abs/2307.09009

Noam Chomsky, 1981. *Lectures on Government and Binding.*

Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, 2018. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* https://arxiv.org/abs/1810.04805

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, Dongmei Zhang, 2019.
Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation.

Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, 2013. *Efficient Estimation of Word Representations in Vector Space.* https://arxiv.org/abs/1301.3781

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer, 2018. *Deep contextualized word representations.* https://arxiv.org/abs/1802.05365

K. Schuler, 2006. *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon.*

G.H. von Wright, 1951. Deontic logic. Mind 60 (237):1-15.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, Quoc V. Le, 2019. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. ttps://arxiv.org/abs/1906.08237

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, Dragomir Radev, 2018. *Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task.*

Zhen Wang, 2022. *Modern Question Answering Datasets and Benchmarks: A Survey.* https://arxiv.org/abs/2206.15030

Victor Zhong, Caiming Xiong, Richard Socher, 2018. *Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning.*

FF2020

Flying Forward 2020 (FF2020) is a collaborative three-year research and innovation project funded by the European Union under the Horizon 2020 programme. Our project is developing an entire state-of-the-art Urban Air Mobility (UAM) infrastructure by incorporating this new form of mobility within the geospatial digital infrastructure of cities. It includes a governance model and framework, a regulatory framework, a geospatial digital infrastructure, a Digital Toolbox, an Identity of Things (IDoT) scheme, and interoperability frameworks. The solutions developed during the project will be tested in five living labs across Europe: Eindhoven, Milan, Zaragoza, Tartu and Oulu – enabling an open dialogue with stakeholders, end-users and citizens to improve processes and results. Ultimately, our goal is to have a positive and lasting impact on the quality of life of European citizens and to create sustainable partnerships and cities.

Flying Forward 2020 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no.101006828.