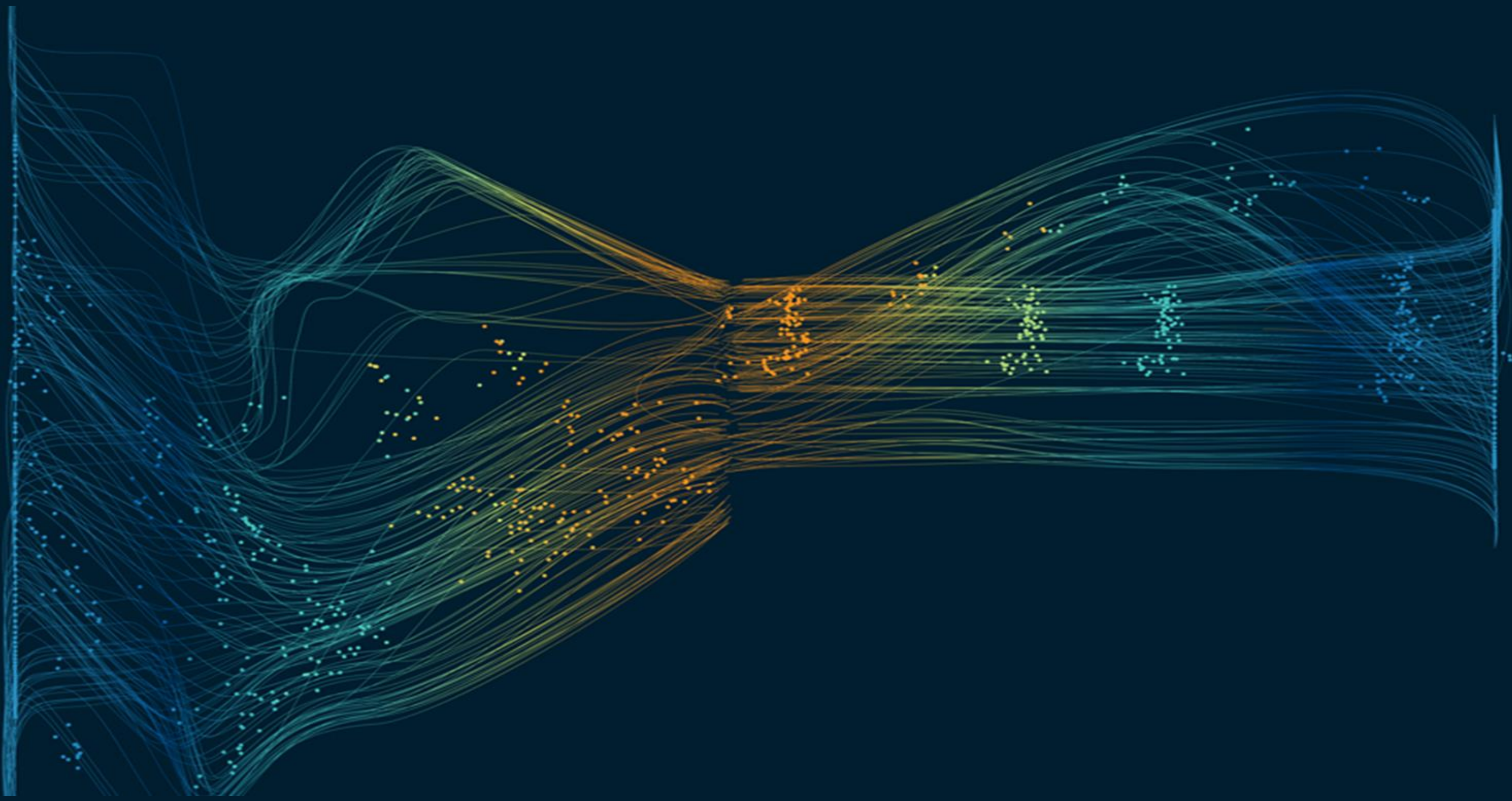
Natural Language Processing for Public Services



DIGIT

Directorate-General for Informatics

DISCLAIMER

This document is for informational purposes only and the Commission cannot be held responsible for any use which may be made of the information contained therein. References to legal acts or documentation of the European Union (EU) cannot be perceived as amending legislation in force or other EU documentation.

The document contains a brief overview of technical nature and is not supplementing or amending terms and conditions of any procurement procedure; therefore, no compensation claim can be based on the contents of the present document.

The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the European Commission. The European Commission does not guarantee the accuracy of the data included in this document. Neither the European Commission nor any person acting on the European Commission's behalf may be held responsible for the use which may be made of the information contained therein.

EUROPEAN COMMISSION

Directorate-General for Informatics
Directorate D — Interoperability Solutions for public administrations, businesses and citizens
Unit D2 — Interoperability Unit

Contact: DIGIT-INTEROPERABILITY@ec.europa.eu

Miguel Alvarez Rodriguez - Project Manager of the Catalogue of Services project

European Commission
B-1049 Brussels

This is a DEP funded report under the Interoperable Europe initiative.

EN PDF ISBN 978-92-76-49263-4 doi: 10.2799/304724

The icons appearing in this study are from https://www.flaticon.com/

DOCUMENT METADATA

Property	Value
Release date	March 2022
Status	Published
Authors	Florian Barthélemy (PwC EU Services) Nathan Ghesquière (PwC EU Services) Nicolas Loozen (PwC EU Services) Louis Matha (PwC EU Services) Emidio Stani (ArchiTES)
Approved by	Miguel Alvarez Rodriguez (Project manager at DG DIGIT)
Keywords	Natural Language Processing, Single Digital Gateway (SDG), Automatic tagging, Public Service, Public Administration, Language model

Table of Contents

1. Introduction - Natural Language Processing to leverage unstructured data

This report is written in the context of the Catalogue of Services action part of the former ISA2[1] - now Interoperable Europe[2] - initiative. The action supports public administrations in building their digital catalogues of public services, to allow citizens, businesses and public administrations across Europe to access and understand the information they need. The Action provides this support at different levels (international, national, regional, and even local) and in a variety of contexts (Single Digital Gateway, cross-border, national harmonisation and other specific implementations). This study is part of the collection of publications[3] in the programme aiming at exchanging knowledge, making service descriptions fit for exchange and providing advice in adopting standards and technologies.

Natural Language Processing (NLP), is the sub-field of artificial intelligence[4] focusing on the interaction between computers and human language. It is defined as "*the machine's ability to identify, process, understand and/or generate information in written and spoken human communications*" (AIWATCH, 2020)[5]. The purpose of this technology is to make machines capable of reading and reasoning with human language and therefore, automatically process it. Some common tasks involving NLP include information extraction, document categorization and semantic text matching.

While NLP is not a new science, the growing capabilities of algorithms, the exponential availability of data and the enhanced computing power have taken this technology to an unprecedented level of adoption. Today, NLP is widely diffused, from the virtual assistant[6] to Google Translate[7].

The public sector is also impacted by this technology. Public organisations often lack structure which can make it difficult to interpret the available documents and data. NLP could help them leverage this data by automatically processing them. Using NLP, public organisations can improve their effectiveness to reallocate some of their time to higher value-added activities or improve the quality of their services by deriving new insights out of texts.

1.1. Scope and objectives

This document aims at identifying how NLP can be leveraged by public organisations. In doing so, guidelines and good practices are provided for public organisations who are interested in starting an NLP project. Therefore, the primary audience of this study are decision-makers, product or service owners or more generally, people with a business background. By reading this document, the reader should be able to answer questions such as:

[1] https://ec.europa.eu/isa2/isa2_en

[2] https://joinup.ec.europa.eu/collection/interoperable-europe/interoperable-europe

[3] https://joinup.ec.europa.eu/collection/interoperable-europe/publications

[4] According to the Computer Science Ontology: https://cso.kmi.open.ac.uk/topics/natural_language_processing

[5] https://publications.jrc.ec.europa.eu/repository/bitstream/JRC118163/jrc118163_ai_watch._defining_artificial_intelligence_1.pdf

[6] https://link.springer.com/chapter/10.1007%2F978-981-13-5758-9_17

[7] https://en.wikipedia.org/wiki/Google_Neural_Machine_Translation

- What is NLP? (section 1)
- What are the main application areas of NLP? (section 2)
- What should my organisation know before investing in NLP? (section 3)
- What is the generic process for implementing NLP? (section 3)
- What are existing use cases and success stories in the public sector around NLP in Europe? (section 4)
- How can NLP techniques be used for automated tagging of HTML pages? (section 5)

Through the means of desk research and interviews with relevant stakeholders who conducted NLP projects in the public sector, an overview is given on NLP application areas, model creation, concrete examples of NLP and a Proof-of-Concept on automatic tagging using NLP.

Throughout the document, several more in-depth technical analyses are also provided to demonstrate how an NLP model can be built in a more hands-on way. These are included to enable organisations to take the steps necessary to build their own NLP model, but are by no means obligatory to have a basic understanding of what NLP can bring to a public organisation. These sections are indicated as follows:

The content of this section is technical and provides technical insights to build the Language Model

2. Application areas for NLP

In the public sector, NLP can be deployed in several different ways and for a variety of different purposes. Below are several of the most common application areas where NLP algorithms are used in practice[8] [9]. This list represents a general overview of application areas and is therefore not unique to the public sector. For each application area, a short description is provided followed by concrete examples. Afterwards, section 3.4.1 will deep-dive into the algorithms and models that can be used in relation to specific application areas.

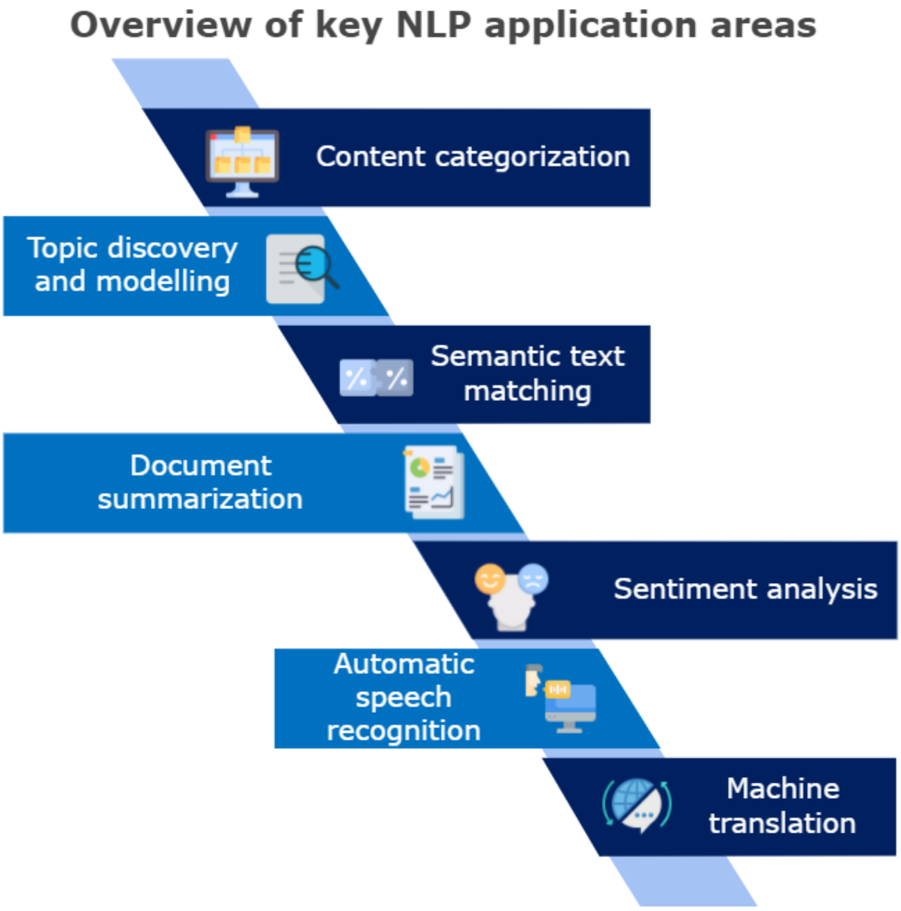


Figure 1: Overview of key NLP application areas

[8] https://www2.deloitte.com/us/en/insights/focus/cognitive-technologies/natural-language-processing-examples-in-government-data.html

[9] https://research.aimultiple.com/nlp-use-cases/

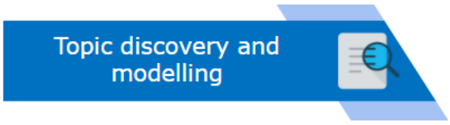
2.1. Content categorization

Content categorization is a fundamental application area of text classification algorithms. Text classification is an NLP machine learning technique that allocates a set of predefined categories to text. By using such models, entire (unstructured) texts can be classified into predefined categories, allowing effective content categorization. Since the categories are identified upfront, this is a supervised machine learning method (for more information on the difference between supervised and unsupervised methods, see section 7). Practical examples of content categorization in practice include the organisation of articles by topic, automation of end-user support processes, Search Engine Optimization (SEO) and more.

Example: *Case classification to remove guesswork in populating case fields*[10]

An example of content categorization in practice can be found in case management during the CRM process. By applying classification models to historical data of closed cases, predictions can be made, and patterns can be discovered. This allows multiple fields to be filled in automatically (e.g., case reason, language, escalated, priority[11]) so that agents can resolve cases quickly and efficiently.

2.2. Topic discovery and modelling



Topic modelling is the process of automatically identifying topics present in a corpus of texts and to derive hidden patterns present in this corpus. In opposition to content categorization, topic modelling is an unsupervised method since it is the machine itself that defines the topics. Therefore, no labelling is needed for a topic discovery task. This technique is particularly useful for document clustering, information retrieval from unstructured text and feature selection. Example of use cases of topic modelling includes automatically identifying the topic of news articles[12] or predicting trends of scientific research topics[13][14].

Example: *Tracking geographical locations using a geo-aware topic model for analysing social media data*[15]

Researchers developed a tool that can track online discussions geographically using Latent Dirichlet allocation (LDA), which is a method in topic modelling and NLP. The tool was tested during the American presidential elections in 2016 by scanning tweets (unstructured text) to extract the geographical area of the tweet and compare it to actual election locations. This can provide public organisations with useful information about where and which resources to allocate during the elections.

[10] https://help.salesforce.com/s/articleView?id=sf.cc_service_what_is.htm&type=5

[11] https://www.springml.com/blog/einstein-case-classification/

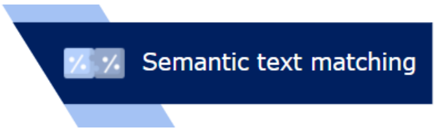
[12] https://www.kaggle.com/rcushen/topic-modelling-with-lsa-and-lda

[13] https://www.datasciencecentral.com/topic-modeling-algorithms-techniques-and-application/

[14] https://www.researchgate.net/publication/327901659_Towards_Predicting_Trend_of_Scientific_Research_Topics_using_Topic_Modeling

[15] https://www.sciencedirect.com/science/article/pii/S0167923617300842

2.3. Semantic text matching [16]



Semantic text matching is the task of estimating the similarity between the source and the target text pieces. Since the versatility of the language makes it difficult to use rule-based methods, deep neural networks are currently used. It is one of the most important research problems since it has applications in information retrieval, question answering and recommendation systems. This task is for instance one of the key components of a search engine[17] since the user wants the web pages to be about the same subject as their query. Of course, use cases using semantic text matching are as various as information extraction to search on long-form documents[18].

Example: *Link e-government services between each other*[19]

Researchers of the University of Milano-Bicocca created a tool using semantic text matching to help public administrations to connect their e-gov services to services, provided by other administrations, already connected to the Linked Open Data cloud.

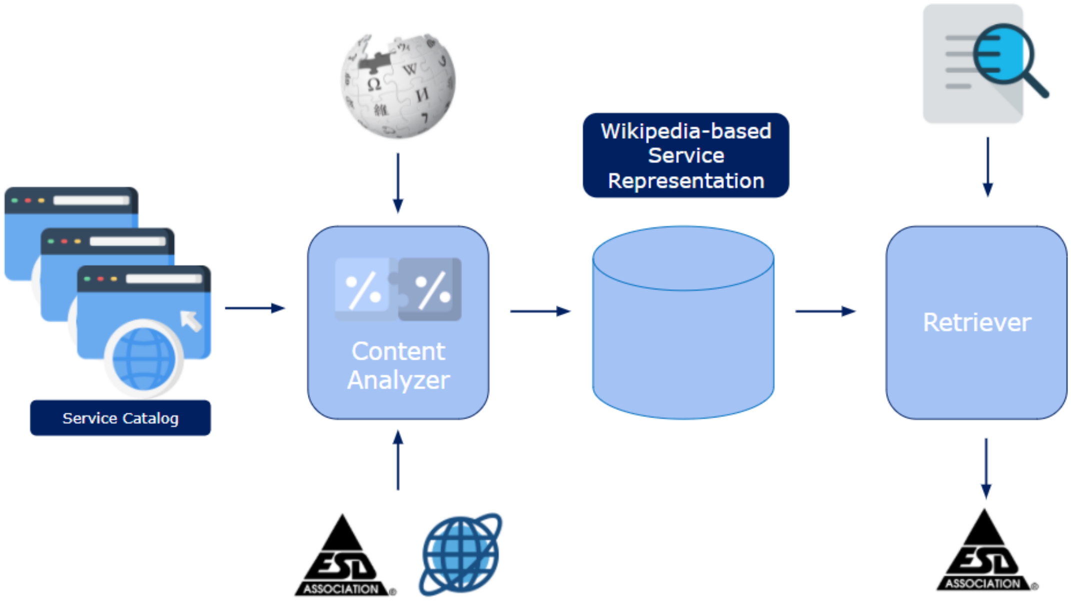


Figure 2: Semantic text matching architecture

2.4. Sentiment analysis

Comparable to content categorization models, sentiment analysis (SA) applications are also based on text classification. Also referred to as ‘opinion mining’, these models use NLP to determine whether the content of data is negative, neutral, or positive. Several types of SA exist, including fine-grained SA, emotion detection, aspect-based SA, multilingual SA and more. Sentiment analysis is highly topical in the world of social media, where companies can track their brand reputation or customer feedback in an automated fashion. Furthermore, companies can analyse thousands of reviews to find what the key pain points are, or track

[16] https://www.sciencedirect.com/science/article/pii/S0167923617300842

[17] https://www.theleverageway.com/blog/google-semantic-search/

[18] https://jyunyu.csie.org/docs/pubs/www2019paper.pdf

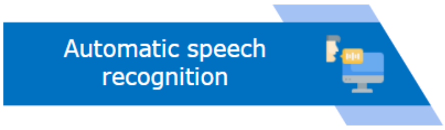
[19] http://ceur-ws.org/Vol-992/paper4.pdf

twitter feeds real-time to quickly react to customer complaints[20]. Sentiment analysis can also be leveraged by chatbots to create interaction with end users, as described in a previous study (section 7.1.1.)

Example: *Tracking of internal social network to interpret employee sentiment*[21]

IBM uses sentiment analysis on their internal social-networking platform to track employee concerns and develop programs to improve employee satisfaction. This allows human-resource staff to automatically analyse large amounts of data and improve the likelihood of retaining their employees[22].

2.5. Automatic speech recognition



Automatic speech recognition (ASR) is the application of NLP which converts spoken words into computer texts. Speech recognition models can be divided into two types[23]: acoustic models and language models. While acoustic models map the sounds signals to phonetic representations, language models capture the domain knowledge of words, grammar, and sentence structure for the language. Use cases of ASR involves virtual assistant such as Siri[24] from Apple, Alexa[25] from Amazon or Google Assistant[26] from Google. ASR can also be used for documentation purposes where the Subject Matter Expert provides a spoken input and the machine automatically edits a document[27]. As mentioned in the report “Architecture for Public Service Chatbot” commissioned by the former ISA^2 Programme (now Interoperable Europe[28]), Automatic Speech Recognition could be used by chatbots to provide alternative ways to interact with end-users.

Example: *Speech recognition to improve productivity of the police*[29]

The Police in San Bernardino, California are using a speech recognition tool to write reports. This tool doubled the efficiency of the creation of reports since the police department averaged 115 words per minute dictating, while the average person types 40 to 60 words per minute.

2.6. Document summarisation

Document summarisation is the NLP method relying on keyword extraction[30] enabling machines to condense a piece of text to a shorter version without losing essential information. Since manual text summarisation can be very time consuming, this application of NLP has grown in popularity over the years with application in legal texts summarisation, news summarisation or

[20] https://monkeylearn.com/sentiment-analysis/

[21] https://www.theatlantic.com/technology/archive/2016/09/the-algorithms-that-tell-bosses-how-employees-feel/502064/

[22] https://www.ciodive.com/news/companies-using-sentiment-analysis-software-to-understand-employee-concerns/407357/

[23] https://www.analyticsvidhya.com/blog/2021/01/introduction-to-automatic-speech-recognition-and-natural-language-processing/

[24] https://www.apple.com/fr/siri/

[25] https://en.wikipedia.org/wiki/Amazon_Alexa

[26] https://cloud.google.com/speech-to-text

[27] https://en.wikipedia.org/wiki/Speech_recognition#Applications

[28] https://joinup.ec.europa.eu/collection/interoperable-europe/interoperable-europe

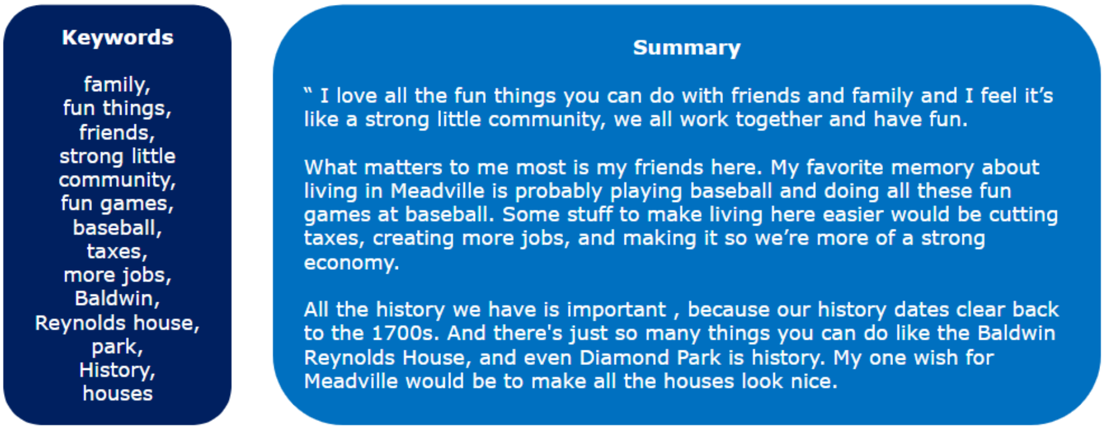
[29] https://gcn.com/articles/2019/12/12/voice-recognition-police-reports.aspx

[30] https://arxiv.org/ftp/arxiv/papers/1704/1704.03242.pdf

headline generation. There are two different approaches in text summarisation: the extractive approach and the abstractive approach. The extractive approach picks up sentences directly from the to-be-summarised document to create the summary. This approach identifies the key sentences of the document and puts them together to produce a condensed version. The abstractive approach creates a new text summarising the to-be-summarised document using advanced NLP tools to understand it. Since this approach is more challenging than the other one, it takes advantage of the latest breakthrough in NLP such as Recurrent Neural Networks[31].

Example: *Enhancement of Public Services in Meadville using text summarisation*[32]

The city of Meadville in Pennsylvania has conducted interviews of their residents to understand what their concerns were and transcribed these interviews into text files. Text summarisation was used to highlight the key insight of these interviews and saved over 300 hours of human labour.



Keywords

family,
fun things,
friends,
strong little community,
fun games,
baseball,
taxes,
more jobs,
Baldwin,
Reynolds house,
park,
History,
houses

Summary

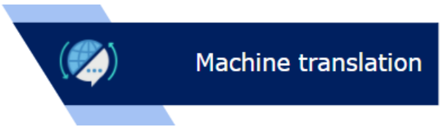
“ I love all the fun things you can do with friends and family and I feel it’s like a strong little community, we all work together and have fun.

What matters to me most is my friends here. My favorite memory about living in Meadville is probably playing baseball and doing all these fun games at baseball. Some stuff to make living here easier would be cutting taxes, creating more jobs, and making it so we’re more of a strong economy.

All the history we have is important , because our history dates clear back to the 1700s. And there's just so many things you can do like the Baldwin Reynolds House, and even Diamond Park is history. My one wish for Meadville would be to make all the houses look nice.

Figure 3: An example of document summarisation

2.7. Machine translation



Machine translation (MT) is an application area that uses NLP to translate text from one language to another. Several different types of machine translation exist, but most are based on linguistic rules where the to-be translated word is replaced by the most accurately corresponding target word. On a fundamental level, MT algorithms execute a mechanical substitution of words between languages, which means that they create a symbolic representation of sentences which are then decoded into the target language. Types of MT models include rule-based MT, statistical MT, example-based MT, hybrid- and neural MT. Machine translation is applied in the creation of online translation apps, B2B domain-specific translation, speech-to-speech translation, and other areas.

Example: *The website of the Fairfax County Public Schools*[33]

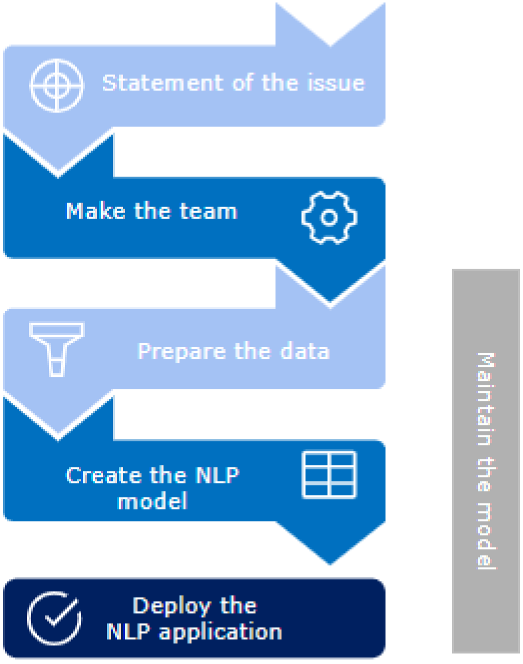
This website uses Google Translate to provide information in 90 languages, from Afrikaans to Zulu. It enables a better inclusivity of all the citizens since everyone, regardless of language, can have access to the information.

[31] https://arxiv.org/pdf/1706.03762.pdf

[32] https://arxiv.org/pdf/1910.10490.pdf

[33] https://www.fcps.edu/

3. Structure of an NLP project



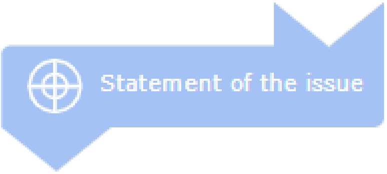
This section gives an overview of the practical steps that can be undertaken to structure an NLP project.

Several steps are discussed, including defining the problem, building the team, preparing the data, creating the model, deploying the model, and maintaining it. These steps are common for NLP projects and can be found in multiple other reports on NLP[34][35]. The added value of this report is the depth of the analysis for each step and the content that is targeted for the public sector.

Note that these steps are not fixed nor universal for each NLP problem. They primarily serve as a foundation on which to build further.

Figure 4: structure of an NLP project

3.1. Statement of the issue



The first step is to define the problems public organisations face and what is the scope of the project. Based on this definition, the team can be built, and the technical requirements can be set.

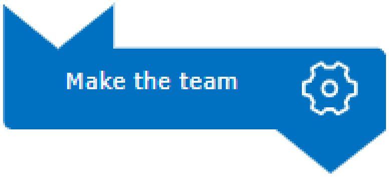
After such work, the data to be processed can be identified and infrastructure can be drawn. Of course, the infrastructure will depend also on the current IT components within the organisation: the organisation will want to reuse what it already has. Usually, the IT infrastructure of the project is set up before processing the data.

The business definition of the problem has a huge impact on the data processing. The best practice is to link the business definition to the NLP capabilities (defined in section 2).

[34] https://medium.com/modern-nlp/productionizing-nlp-models-9a2b8a0c7d14

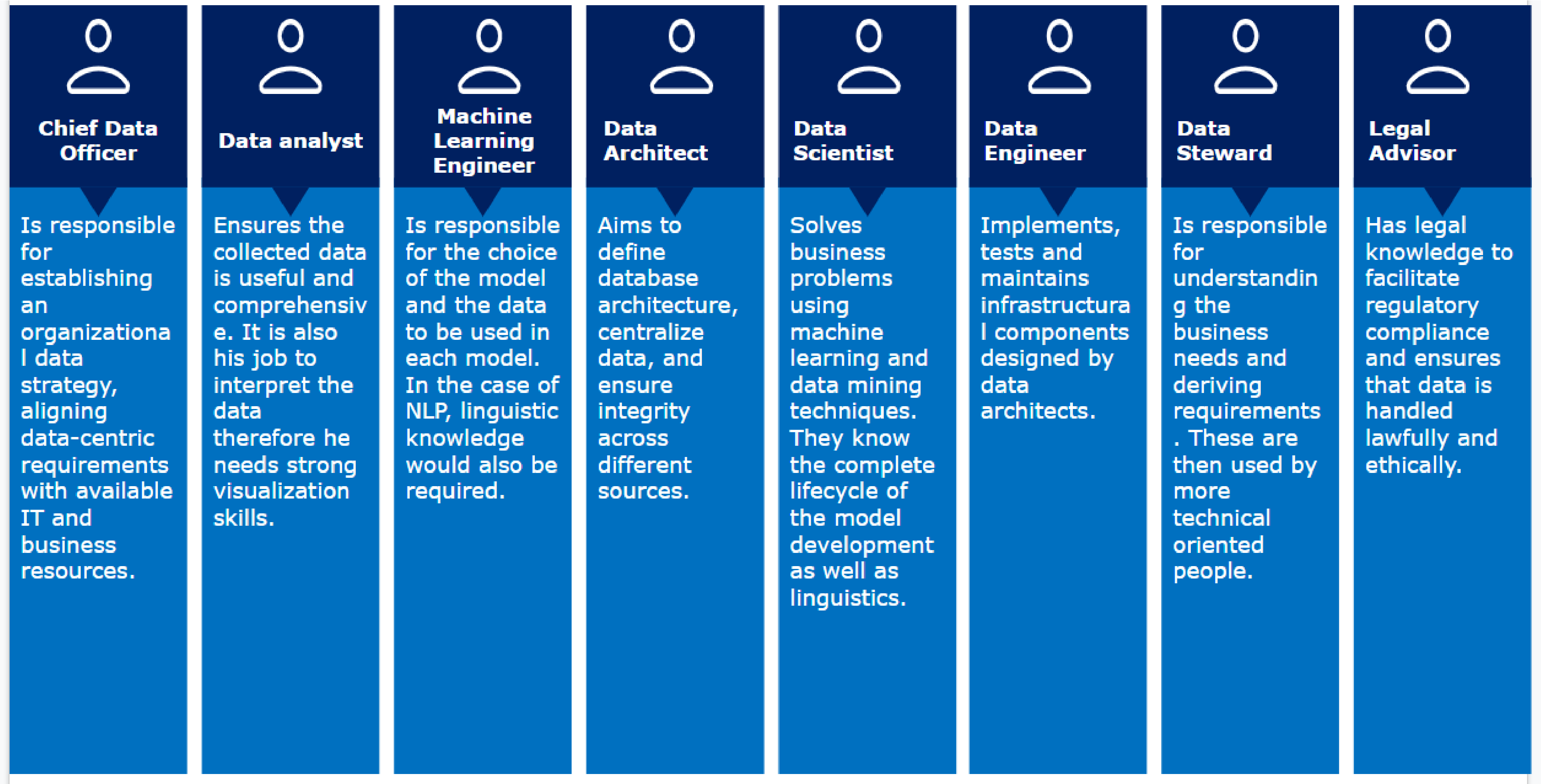
[35] https://www2.deloitte.com/us/en/insights/focus/cognitive-technologies/natural-language-processing-examples-in-government-data.html

3.2. Make the team



To launch an NLP project, a public organisation needs to have the right skill set in-house or to hire them (at least for the time of the project). Of course, the people needed may vary depending on the scale and the complexity of the project. Since the pace at which data science and natural language processing fields are evolving drastically, there are no strict roles defined for a data science project yet. Furthermore, the roles needed for an NLP project depend highly on the scope and the nature of the project. Therefore, some roles presented below may be overlapping with each other or be relevant only for big NLP projects (e.g., projects for proposing an external service). Here is a sample of the roles that may be needed in an NLP project[36]:

[36] https://www.altexsoft.com/blog/datascience/how-to-structure-data-science-team-key-models-and-roles/



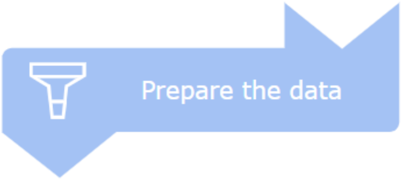
Chief Data Officer	Data analyst	Machine Learning Engineer	Data Architect	Data Scientist	Data Engineer	Data Steward	Legal Advisor
Is responsible for establishing an organizational data strategy, aligning data-centric requirements with available IT and business resources.	Ensures the collected data is useful and comprehensive. It is also his job to interpret the data therefore he needs strong visualization skills.	Is responsible for the choice of the model and the data to be used in each model. In the case of NLP, linguistic knowledge would also be required.	Aims to define database architecture, centralize data, and ensure integrity across different sources.	Solves business problems using machine learning and data mining techniques. They know the complete lifecycle of the model development as well as linguistics.	Implements, tests and maintains infrastructural components designed by data architects.	Is responsible for understanding the business needs and deriving requirements. These are then used by more technical oriented people.	Has legal knowledge to facilitate regulatory compliance and ensures that data is handled lawfully and ethically.

Figure 5: Potential team composition for an NLP project

The above list of roles is by no means exhaustive and some roles may not be relevant for a given project. One should always consider the size, nature, and scope of the NLP project to assess which roles are needed. Each step of the model creation process takes up a certain amount of time from the team involved, depending on the NLP project. Several studies examine the proportional time spent on each step, resulting in useful insights for an organisation wanting to leverage AI. Some studies estimate the time spent on creating the model (section 3.4) to be limited to approximately 15% of total time, with the other tasks taking up most of the human input. More specifically, an average of 20% of the time would be spent on data collection, another 20% on data preparation and the remaining on evaluating, updating, and studying the model[37] [38]. Another source makes similar estimations with data gathering (16,8%), data cleaning (22,9%), data visualisation (13,6%) taking up the most significant share of human effort. This study then found an average of 20,9% for building the model, 9% for putting the model in production and 16.7% for finding insights, communicating with stakeholders and other activities[39]. Again, this will vary significantly depending on the NLP project. Much of the human work today can be simplified because of unprecedented open-source availability. This means that using pre-existing components requires very little human work, whereas building an NLP system from scratch is an exceptionally complex task. Moreover, ample cloud-based platforms exist that have capabilities to quickly and easily build NLP applications. Platforms such as Azure Cognitive Services[40] and DeepNLP[41] allow the use of domain-specific, pre-trained models through an intuitive workflow, enabling even low-experienced users to build meaningful models. A study on enterprise machine learning found that 40% of companies say it takes them over one month to deploy a ML model whereas other studies indicate it usually takes several months or years. Other sources claim that one skilled data scientist could come up with a first working AI prototype in one day[42]. It can be concluded that complexity, uniqueness, and team experience are all factors that will determine the human effort needed to build an NLP model. Using aforementioned cloud platforms such as Microsoft Azure Cognitive Services[43], Amazon Comprehend[44] or Google Cloud Natural Language[45] can also significantly reduce development time through off-the-shelf NLP capabilities without training needed.

In the next sections, we introduce the generic NLP steps in which the data roles identified in the figure are represented. In each step, we have described the responsibilities of these different roles.

3.3. Prepare the data



Next, organisations should identify the relevant data and determine its accessibility. Data is at the foundation of any Machine Learning or Natural Language Processing model, and without data of sufficient volume and quality it is impossible to achieve valuable results. This is an important consideration when building NLP applications.

[37] https://www.researchgate.net/publication/241623879_A_study_on_the_importance_of_and_time_spent_on_different_modeling_steps

[38] https://thenewstack.io/add-it-up-how-long-does-a-machine-learning-deployment-take/

[39] https://businessoverbroadway.com/2019/02/19/how-do-data-professionals-spend-their-time-on-data-science-projects/

[40] https://azure.microsoft.com/en-us/services/cognitive-services/#overview

[41] https://www.sparkcognition.com/products/deepnlp/

[42] https://hackernoon.com/towards-ai-how-long-does-it-take-you-to-go-from-idea-to-working-prototype-a-day-a-month-8a03ffecca0a

[43] https://azure.microsoft.com/en-us/services/cognitive-services/

[44] https://aws.amazon.com/comprehend/

[45] https://cloud.google.com/natural-language

Some data may be easily acquired; some may not be in a machine-readable format or may be unlabelled or of poor quality. During the NLP building process, an important objective is to be able to match words and phrases with similar meanings but coming from different input data[46]. This requires particularly large data sets, as the data is typically unstructured and diverse.

One can significantly improve the result of this matching process by using state-of-the-art NLP pre-processing techniques. Preparing a text means to bring the text into a form that is predictable and analysable for the task. To correctly prepare the data, several key steps need to be done (please note that this method and flow is not the only possible way of preparing data). Many different techniques and processes exist that can be used towards different objectives. The steps in this section are generic for all NLP techniques, but will later be related to specific examples in the public sector.

Relevant profiles for this phase of the NLP pipeline are the following:

- **Data Steward:** the data steward is responsible for defining which data needs to be analysed and what is the final output.
- **Data Analyst:** the data analyst is responsible for collecting the data and pushing it into the right component.
- **Legal Advisor:** the legal advisor is responsible for making sure that data has been collected, stored and used in accordance with legal and ethical guidelines.
- **Data Engineer:** the data engineer is responsible for bringing together the data from different sources by building data pipelines.
- **Data Architect:** the data architect is responsible for designing the IT components that will store the data.
- **Data Scientist:** the data scientist is responsible for cleaning the textual data.

3.3.1. Collect

After the data sources have been identified by the data steward, it can be collected and uploaded in a target database. Sources of data are often heterogeneous, ranging from business systems to Application Programming Interfaces (APIs), to sensor data, marketing tools, transaction databases, and others. For example, different stores from a supermarket chain might have their data stored in databases owned by the individual stores. Each of these stores can have its own database, storing data in a different format, with different rules and points of access (API, bulk download, etc.). Moreover, each database can have different access restrictions, licensing permissions or sensitivity, each influencing the way developers or IT components will be able to access, collect and reuse this data. Consequently, the 'collect' stage of the data preparation will require methods to retrieve the relevant data from each store into a centralised data warehouse.

At this stage, a first consideration of model deployment and architecture should be foreseen as well. Involving the data architect at this stage allows the data preparation to happen in accordance with the structures that will be used to build the actual NLP model at a later stage. This way, it can be made sure that the necessary architecture is operational as the model is being built. A more in-depth analysis of model deployment can be found in section 3.5. This example indicates that the complexity of collecting data heavily depends on the context and stakeholders involved.

[46] https://arxiv.org/pdf/1903.06902.pdf

In summary, important questions to consider during this stage are as follows:

- What are the types of data I need?
- Which data does my organisation have?
- Where is the data stored?
- Can the data be accessed?
- What security or privacy measures need to be considered?
- What are the estimated efforts for all these considerations?

3.3.2. *Transform*

When the data from various sources is identified and collected, it needs to be transformed. The objective of this step is transforming the raw data that has been extracted from the sources into a format that can be used by different applications.[47] More concretely, various formats such as texts, images, graphs, time series etc. are transformed to a target format that can be used during the following stages of the NLP building process. Transforming data from one format to another might lead to information loss (e.g., from xml to json there is a loss of the names pacing thus entities look like they are in the same namespace while they might conflict). This operation must be carefully designed as syntaxes are different and naming rules must be put in place.

3.3.3. *Clean*

Once the data is transformed into the right format, it is followed by a cleansing procedure. Texts usually contain many insignificant words and characters that are not needed when training an NLP model. The cleaning process addresses this by removing incorrect, incomplete, irrelevant, and outdated data. Furthermore, many methods exist to augment or group text contents, improving the performance of the natural language processing algorithm. This step is usually performed by the data scientist, possibly aided by an NLP platform as mentioned before. Several examples of such techniques are displayed in the table below.

Pre-processing steps	
Spellcheck	Raw user input data can contain spelling errors. A spell checker can be used here.
Split into sentences	It could be helpful to analyse every sentence separately. NLP libraries (e.g., NLTK60[48], StandfordNLP61[49], SpaCy62[50], KNIME[51]) can be used to split text into sentences.

[47] https://databricks.com/glossary/extract-transform-load#:~:text=ETL%2C%20which%20stands%20for%20extract,downstream%20to%20solve%20business%20problems.

[48] https://www.nltk.org/

[49] https://nlp.stanford.edu/software/

[50] https://spacy.io/

[51] https://hub.knime.com/knime/extensions/org.knime.features.ext.textprocessing/latest

Upper/lower case	Converting the entire text into uppercase or lowercase, so that the algorithm does not treat the same words differently.
Removing noise & stop words	Everything that is not a standard number or letter can be removed. Also, very common words which would appear to be of little value for the model are excluded from the vocabulary entirely. These stop words can then safely be removed.
Split into words	Hardcoded rules typically operate with words. This can be done with the NLP libraries mentioned above.
Part-of-speech (POS) tagging	Some words have multiple meanings or can be both a noun and a verb if seen separately. POS tagging is the process of linking a word with a particular part of speech ('adverb', 'noun', 'verb'…), based on context and definition. This way, the meaning in a specific context can be derived. The same NLP libraries can be used as previously mentioned, or Google SyntaxNet63[52], which is more accurate and supports multiple languages. Furthermore, ISA2[53] (now Interoperable Europe[54]) and DEP[55] semantic models that define terms in a specific context or application area could be valuable for this purpose, as they provide knowledge about meaning and relations of words[56] [57] (see also section 4.3.2).
Lemmatise words	One word can have many forms: e.g., 'pay', 'paying' and 'paid'. In many cases, the exact form of the word is not important for writing a hardcoded rule. If pre-processing code can identify a lemma, a canonical form of the word, it helps to simplify the rule. Lemmatisation, identifying lemmas, is based on dictionaries, which list all forms of every word. The most popular dictionary for English is WordNet64[58]. Furthermore, the ISA^2 and DEP vocabularies can be relevant in the lemmatisation process for public organisations as explained above.

[52] https://opensource.google.com/projects/syntaxnet

[53] https://ec.europa.eu/isa2/home_en

[54] https://joinup.ec.europa.eu/collection/interoperable-europe/interoperable-europe

[55] https://digital-strategy.ec.europa.eu/en/activities/digital-programme

[56] https://aclanthology.org/W04-0609.pdf

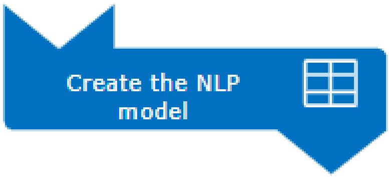
[57] https://joinup.ec.europa.eu/collection/semantic-interoperability-community-semic/our-solutions

[58] https://wordnet.princeton.edu/

Stemming	Like lemmatisation, stemming aims to reduce different forms of a word to a common base form. However, stemming involves a cruder process of ‘chopping off’ the ends of words to achieve a more simplified form. Contrary to lemmatisation, it does not make use of vocabularies to reduce words to a base form[59].
Entity recognition	Dates and numbers can be expressed in different formats: “3/1/2016”, “1st of March”, “next Wednesday”, “2016–03–01”, “123”, “one hundred”, etc. It may be helpful to convert them to a unified format before doing pattern matching. Other entities which require special treatment: locations (countries, regions, cities, street addresses, places), people, phone numbers, … For entity recognition, vocabularies, and models within the former ISA2[60] (now Interoperable Europe[61]) can be highly valuable as well to derive context and meaning.

Table 1: data pre-processing techniques

3.4. Create the Language model



Once the required data is available in the right format, structure, and level of quality, one can start building the Language model[62]. This step is one of the most difficult and crucial to the performance of the application: The Language model understands and processes the textual data. The algorithms able to convert texts into vectors are called Word Embedding or Language Models. This section will explain with more detail what word embeddings are and how to obtain them. Next, an introduction to the application layer of an NLP model is provided.

First, excluding the probabilistic type of language models, the report will focus over NN (neural network based modern language models. Relevant profiles for this phase of the NLP pipeline are the following:

- **Data Scientist:** the data scientist is responsible for creating the NLP model alongside the machine learning engineer and ensuring it is consistent with the data we have.
- **Machine Learning Engineer:** the machine learning engineer is responsible for creating the NLP model alongside the data scientist and ensuring it can be developed at scale.

[59] https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

[60] https://joinup.ec.europa.eu/collection/semantic-interoperability-community-semic/our-solutions

[61] https://joinup.ec.europa.eu/collection/interoperable-europe/interoperable-europe

[62] A language model is basically a probability distribution over words or word sequences. In practice, a language model gives the probability of a certain word sequence being “valid”. From Mor Kapronczay. Source: https://towardsdatascience.com/the-beginners-guide-to-language-models-aa47165b57f9

- **Data Architect:** the data architect is responsible for building the IT component necessary to develop the NLP model.

3.4.1. Word embeddings

The content of this section is technical and provides technical insights to build the Language Model

When data has been collected and prepared, it needs to be converted into vectors called “embeddings”. These embeddings will be then processed by the machine learning engineer and data scientist according to the goal of the NLP pipeline. This step is one of the most crucial in the NLP project: the language model is the component that understands the human language. The quality of the language model will have a huge impact on the quality of the output. In the following section there will be a comparison between the 3 maybe most used language models.

	Word to Vector (Word2Vec)	Global Vector (GloVe)	Bidirectional Encoder Representation from Transformers (BERT)
General description	Word2vec[63] (Word to Vector) is a technique for natural language processing published by 2 researchers at Google in 2013 that uses a neural network model to learn associations from a large corpus of text. Word2vec represents each distinct word with a list of numbers called a vector. These vectors are chosen so that a simple mathematical function (the cosine similarity) can indicate the level of similarity between the words represented by those vectors.	GloVe (Global Vector) is an unsupervised learning algorithm for obtaining vector representations for words made by Stanford researchers in 2014. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.	BERT (Bidirectional Encoder Representation from Transformers) is a contextual model created by Google researchers in 2019. This word embedding algorithm is the one currently used by Google in its search engine. BERT[64] builds on top of a number of clever ideas that have been bubbling up in the NLP community recently – including but not limited to Semi-supervised Sequence Learning (by Andrew Dai and Quoc Le), ELMo (by Matthew Peters and researchers from AI2 and UW CSE), ULMFiT (by fast.ai founder Jeremy Howard and Sebastian Ruder), the OpenAI transformer (by OpenAI researchers Radford, Narasimhan, Salimans, and Sutskever), and the Transformer (Vaswani et al)
Technical description	Two architectures can be used to produce a distributed representation of words: continuous bag-of-words (CBOW) or continuous skip-gram. According to the author's note, CBOW is faster while skip-gram is slower but does a better job for infrequent words. The model is explained in the three research papers [65]made by Google researchers.	The difference between GloVe and Word2Vec is that Word2vec embeddings are based on training a shallow feedforward neural network while glove embeddings are learnt based on matrix factorisation techniques. However, like Word2Vec it uses context to understand and create the word representations. In practice, GloVe is not better than Word2Vec in all use cases therefore, one needs to test which one performs better.	To understand the key characteristics of BERT, the classification of Word Embedding algorithms should be explained first. They are divided into two main categories, context-free and contextual. Context-free models (such as Word2Vec or GloVe) generate a single word embedding representation for each word in the vocabulary. For example, the word "bank" would have the same context-free representation in "bank account" and "bank of the river".

[63] https://code.google.com/archive/p/word2vec/

[64] https://jalammar.github.io/illustrated-bert/

[65] Three research papers describing word2vec architecture: https://arxiv.org/pdf/1301.3781.pdf ; ttps://arxiv.org/pdf/1310.4546.pdf ; https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/rvecs.pdf

			Contextual models generate a representation of each word that is based on the other words in the sentence. These types of models fall into two categories, they can be either unidirectional or bidirectional. For example, in the sentence “I made a bank deposit” the unidirectional representation of “bank” is only based on “I made a” but not “deposit”. Bidirectional models instead use both their right and left contexts.
Author	Google	Stanford University	Google
Release date	2013	2014	2019
Libraries	Word2Vec[66] is open source licensed under the Apache License 2.0. A permissive license whose main conditions require preservation of copyright and license notices. Contributors provide an express grant of patent rights. Doc2Vec[67] is an improvement of Word2Vec made in 2014 to analyse pages. This library is open source. Fasttext[68] corrects one of the biggest flaws of Word2Vec: if one gives it a word which is not in the vocabulary used to train the model, it cannot give similar words. This library is MIT-licensed and was made in 2016.	GloVe[70] is under open source licensed under Apache License 2.0	BERT[71] is under an Apache-2.0 licence. Currently, more advanced versions of BERT are available: DistilBERT is a model created by researchers of HuggingFace in 2020. This model has almost the same performance as BERT while being 40% lighter and 60% faster. RoBERTa is an improved version of BERT created by Facebook researchers in 2019 that achieves better accuracy by changing the neural network architecture. The model used in the research paper is available on GitHub. The content of the

[66] Link to the Word2Vec project: https://code.google.com/archive/p/word2vec/

[67] Research paper of Doc2Vec : https://arxiv.org/pdf/1405.4053.pdf

[68] Link to the library: https://fasttext.cc/

[70] Link to the GloVe project: https://nlp.stanford.edu/projects/glove/

[71]Github repository of the Transformers library : https://github.com/huggingface/transformers

	Rdf2Vec [69] is a tool for creating vector representations of RDF graphs. It uses the same architecture as Word2Vec and allows learning patterns about entities in the graph.		repository is under an MIT licence. XLNet is the latest improvement of BERT to improve the accuracy of the word embedding. It was designed by Google researchers in 2020. In the end, XLNET outperformed BERT on 20 tasks, often by a large margin including question answering, natural language inference, sentiment analysis and document ranking. The model used in the research paper is available on GitHub under an Apache-2.0 licence.

Table 2: Language model overview

[69] Rdf2Vec: http://rdf2vec.org/

Overall, the choice of the language models will depend on multiple factors.

There are two types of Neural Network based language models: pre-trained and to-be-trained. The difference between the two relies on whether the language model has been trained beforehand or not. One should know that a language model needs to be trained on text before being used. This is an important decision point in an NLP project. Therefore, we describe several options below which can be considered at this point.

The to-be-trained models are language models that need to be trained within the scope of the project. It requires time, computing resources and a huge amount of text to be trained; however, the understanding of the text to be processed by the language model will be targeted to the project needs leading to better results. Training a complex language model (such as BERT) requires a specific infrastructure, leading to an increase of the cost of the project. The pre-trained models are off-the-shelf language models ready to be used because they have already been trained. Compared to the to-be-trained language models, it is easier to use them on a project and does not (usually) require a specific infrastructure to be used. However, their understanding of the text to be processed can be reduced which could lead to poor results. Naturally, one can also use a pre-trained model (trained on a large corpus of text) and further finetune it with the data available. Such approach has several benefits[72]: (i) Pre-training on the huge text corpus can learn universal language representations and help with the downstream tasks; (ii) Pre-training provides a better model initialisation, which usually leads to a better generalisation performance and speeds up convergence on the target task; (iii) Pre-training can be regarded as a kind of regularisation to avoid overfitting on small data.

Among the three language models described above, all of them are suited for all the NLP tasks. The difference between them relies on the size of the model (BERT is significantly bigger than Word2Vec or GloVe). Usually, in NLP, the bigger the language model, the better. However, a huge language model will require a specific infrastructure to be trained and will take more memory. The performance and size of Word2Vec and GloVe are usually equivalent: depending on the project, one might be better than the other. Therefore, data scientists usually test both to know which one is better suited for a specific task.

3.4.2. Application layer

After applying a language model to transform raw text into embeddings, the next step is to parse the embeddings according to the NLP application of our project (defined in section 2). Since this part of the creation of the NLP model is highly dependent on the NLP application, two different examples will be presented: Text Classification and Named-Entity-Recognition (NER). Text classification is used as a basis for the Proof-of-Concept attached to this study (chapter 5) and NER is one of the most widely used applications of NLP. The table below provides a description, key comparative highlights, and relevant application areas for each of the two methods.

	Text Classification	**Named-Entity-Recognition**
Description	Text classification is a machine learning technique that allocates a set of predefined categories to open-ended text. Text classifiers can be used to structure, organise, and	Named entity recognition (NER) is a natural language processing technique that can automatically scan entire articles and pull out some fundamental

[72] https://www.researchgate.net/publication/340021796_Pre-trained_Models_for_Natural_Language_Processing_A_Survey?__cf_chl_captcha_tk__=D36CR.huIobOgQXPeHogzCnvB86AwMnwF4x5lxCMiYs-1641990584-0-gaNycGzNDD0

	categorise many kinds of text – documents, studies, and files all over the web[73]. Many techniques exist to classify texts using machine learning. Several of the most used examples are listed here (there are more information in section 7.2): - Multinomial Naive Bayes. - Support Vector Machines (SVM). - Random forests. - Neural Networks.	entities in a text and classify them into predefined categories[74]. It is the process of recognising different types of named entities in unstructured texts. Examples of such entities can be person names, location names, company names etc.
Key comparative highlights	- Classification of entire pages or texts. - Allocates using predefined categories.	- Operates at word-level. - Heavily relies on hand-crafted attributes and domain-specific information.
Application Areas	- Content categorization. - Sentiment analysis.	- Content categorisation - Machine Translation[75].

Table 3: Application layer overview

3.5. Deploy the NLP application

Deploy the NLP application

For a machine learning model to deliver its full value by interacting with other components (exp: a website), it needs to be deployed in a specific solution. Then, the environment will use the model via a simple API call. To deploy the NLP model, one should use the techniques deployed below.

To deploy a machine learning model, the team can use two solutions[76]: serverless or container. Choosing the good infrastructure will enable an NLP project to have a reliable tool at a reasonable cost.

Relevant profiles for this phase of the NLP pipeline are the following:

- **Data Architect:** the data architect is responsible for designing the IT components responsible for deploying the NLP model.
- **Data Engineer:** the data engineer is responsible for implementing the IT components designed by the data architect to deploy the NLP model.

3.5.1. Serverless

The serverless architecture is the simplest one. It allows developers to focus on writing the code while leaving to the chosen platform the hardware and scaling issues. A serverless environment creates microservices by only writing source code as a series of functions or simple containers. This simple

[73] https://monkeylearn.com/text-classification/

[74] https://www.analyticsvidhya.com/blog/2021/06/part-10-step-by-step-guide-to-master-nlp-named-entity-recognition/

[75] Usage of Named Entity recognition on Machine Translation: https://nlp.stanford.edu/courses/cs224n/2010/reports/singla-nirajuec.pdf

[76] https://www.phdata.io/blog/the-ultimate-mlops-guide-how-to-deploy-ml-models-to-production/

architecture is then deployed to automatically scale servers using a load balancer. Usually, a serverless architecture can be easily set up from the cloud.

The big advantage of the serverless architecture is that it allows a better time-to-market and it does not require a lot of knowledge in ML deployment. However, this kind of architecture might lead to a higher latency than containers because it requires downloading all the setup each time the platform first executes the application on a new server. In practice it means that the first request on a new server will likely have a higher latency than following requests.

Serverless platforms can also be more expensive compared to running on dedicated infrastructure such as containers. If the application is used frequently, it is generally much more cost-effective to run on dedicated servers of an appropriate size.

In summary, the serverless architecture is particularly useful when creating a minimum viable product, when the demand that the deployed service will generate is unknown and when the traffic is relatively low with long gaps between requests.

3.5.2. Container platforms

Container platforms need expertise to work with them, but they offer a great alternative to serverless architectures and are particularly suited for most machine learning applications including NLP. They can be deployed either on premise or on the cloud; in the Proof-of-Concept described later in this document, it is possible to see an example of a container deployed on AWS cloud.

Containers offer more flexibility on server placement and server management. They are also able to mitigate the issues in serverless environments (like “cold start”) because they better manage storage volumes.

The common challenges when dealing with containers are traffic management (how is it routed and balanced) and server management (server to be scaled up or down).

3.6. Maintain the model

Even after building the NLP model and bringing it to production, the model continues to have its own lifecycle and its performance can decrease over time. AI models are trained using historical data. This means that a model running in a static environment with static data will have the same performance without improving or declining. But models do not usually run-in static environments, simply because the business needs evolve continuously; rather, they’re faced with ever-changing environments and variables. Over time, these changes cause degradation in model performance as the model has no predictive power for interpreting unfamiliar data. This performance decline is called model drift.

To avoid this problem, the NLP/data team needs to retrain the model regularly with recent or more relevant instances of data for it to stay up to date. This is the only way to ensure the model continues to perform or improve as intended. Of course, if the NLP model is always analysing the same data, the model will give the same result over time. In this case, there is no need to maintain it. Since retraining a model is a non-trivial task regarding the effort and money it requires, one should really assess the need of such exercise[77].

Relevant profiles for this phase of the NLP pipeline are the following:

- **Machine learning engineer:** the machine learning engineer is responsible for maintaining the model alongside the data scientist with the insights given by the data steward.
- **Data scientist:** the data scientist is responsible for maintaining the model alongside the machine learning engineer with the insights given by the data steward.
- **Data steward:** the data steward is responsible for giving the business insight to ensure the NLP model is performing the way it should.

There are two main approaches to retrain existing models:

Time-based: This approach aims at retraining the model at a regular interval regardless of how it is performing. It needs a good understanding of how the data is likely to change over time: if the interval between two trainings is too great, the model performance will decline.

Threshold-based: This approach aims at retraining the model when it is needed based on key performance indicators (accuracy, bias …). A broad panel of measurement is needed to detect if a model drift is likely to occur.

3.7. Limitations and biases of NLP

The sections above described the general steps required to build an NLP model from start to finish. NLP procedures (such as building a team) in a public context are similar to private instances of NLP, but several other characteristics exist that shape the potential (and challenges) of NLP in public services[78]. The influx of internal and external data in a public context is often excessive and varied, resulting in a high burden for government employees to effectively process this data. Organising how data is obtained, managed, used and secured is part of the data governance structure within an organisation[79] and is a pertinent concern in the age of big data. It allows organisations to unlock the significant value of data assets by allowing improved data analytics, which leads to better decision making. Moreover, data governance plays a crucial role in regulatory compliance, allowing governments to mitigate the risk of breaching major guidelines[80]. Well-developed data governance also allows governments to deal with transparency about data collection, which fosters citizen involvement and trust. As NLP (and AI more broadly) applications are often complex and use highly diverse input data, it is imperative that these algorithms are "transparent to inspection[81]", so that decisions based on this data can be explained.

Additionally, any decision-maker dealing with NLP/AI deals with the challenging dilemma between securing a citizen's privacy and optimising the efficiency and effectiveness of an algorithm, as more data usually leads to a better outcome. Offering the necessary security and making clear the public benefit

[77] https://www.kdnuggets.com/2019/12/ultimate-guide-model-retraining.html

[78] https://ash.harvard.edu/files/ash/files/artificial_intelligence_for_citizen_services.pdf

[79] https://www.datarobot.com/wiki/data-governance/

[80] https://www.bmc.com/blogs/data-governance/

[81] The Cambride Handbook of Artificial Intelligence: https://books.google.be/books?hl=en&lr=&id=RYOYAwAAQBAJ&oi=fnd&pg=PA316&dq=ethics+of+artificial+intelligence&ots=A1X_xlhExs&sig=zezueX3Al7a5l65WsqmOBZOJsPg&redir_esc=y#v=onepage&q=ethics%20of%20artificial%20intelligence&f=false

that flows from the information can stimulate citizens to share their data, which remains a prominent difficulty for public administrations[82].

When it comes to data availability, a barrier is that most NLP models today are built for common languages such as English, Spanish, or French and many resources are available to train these algorithms. However, some languages, in particular those spoken by small populations or groups with little access to the internet or technology, often go overlooked[83]. For example, building a translation model for one of an estimated 3000 languages in Africa[84] could be challenging as a consequence of the lack of input data. This problem has been addressed partly by new technologies such as multilingual sentence embeddings that can identify and leverage shared characteristics between languages[85]. However, this limitation remains an important research topic towards future NLP improvements. An interviewed expert involved in the development of a large-scale translation tool elaborates that training translation tools from scratch requires an estimated minimum input size of 500 000 sentence pairs, whereas a pre-trained model can be fine-tuned on a domain specific dataset with a sample size range in the hundreds to few thousands[86].

Another limitation is the lack of unifying ontologies and semantic repositories that describe the physical world with the precision needed for practical use. Many knowledge bases are too expensive or never end up in practical, widespread use[87]. To address these limitations, the W3C has created standards such as the Resource Definition Framework (RDF) that define metadata about resources. This improves automated processing of web resources, but more development regarding these issues is ongoing and important towards large-scale deployment of NLP. This is also where initiatives such as the former ISA2[88] (now Interoperable Europe[89])play an important role; by creating and diffusing machine-readable standards of data, opportunities to use data for AI and NLP emerge. Further explanation regarding this topic can be found in the example of MIREL, in section 4.3.

[82] https://joinup.ec.europa.eu/sites/default/files/document/2020-07/jrc120399_Misuraca-AI-Watch_Public-Services_30062020_DEF_0.pdf

[83] https://monkeylearn.com/blog/natural-language-processing-challenges/

[84] https://en.wikipedia.org/wiki/Languages_of_Africa

[85] https://arxiv.org/abs/1812.10464

[86] https://adriangcoder.medium.com/how-much-data-to-you-need-ba834d074f3a

[87] https://www.computerworld.com/article/2797861/the-future-of-natural-language-processing.html

[88] https://ec.europa.eu/isa2/home_en

[89] https://joinup.ec.europa.eu/collection/interoperable-europe/interoperable-europe

4. Relevant examples of NLP usages in the public sector in Europe

Despite the significant estimated potential of AI and NLP in the public sector, we can observe a lack of transformative adoption in governments to date. Furthermore, there is no existing study that can confirm the overall benefits of NLP in the public sector[90]. However, there are numerous use cases and examples existing which have proved the interest of public organisations in the field and the value of NLP in concrete cases. Potential use cases that can be of interest for the public sector are analysing public feedback, improving forensics investigations, enhancing policy analysis, improving regulatory compliance, and building efficient Citizen Digital Assistants. Several concrete projects by European public administrations include "Veripol"[91], an AI system in order to detect false police reports in Spain, the automatic reading and digitisation of land registry requests in Sweden[92], a semantic analyser to find relations among text documents in Slovenia and more[93].

In this chapter, we dive deeper into four specific examples where NLP has achieved valuable results in the public sector. For each example, a context is given first. Next, we elaborate how the specific model or application was developed. Finally, more information is provided on the target audience and/or how the application can be used. These examples aim to show the considerations, advantages, efforts, and results involved in leveraging NLP by public organisations, both on EU and national level.

4.1. Example 1: eTranslation

4.1.1. Context

A first practical example of NLP used in a public context is eTranslation[94]. It is a building block in the Connecting Europe Facility (CEF)[95], a key EU funding instrument to stimulate growth, employment, and competitiveness through direct infrastructure investment at European level. Built from scratch by DG Translation, its main objective is to support European and national public administrations, citizens, and businesses to interchange information across different language barriers. It was initially created because of the need for automated translation of legal documents in multiple languages across the EU. These documents contain specialised jargon and existing machine translation services were not tailored to this type of vocabularies.

eTranslation uses machine translation capabilities that will enable all Digital Service Infrastructures (DSIs) to be multilingual. The performance of machine translation largely depends on the quantity and quality of input data used to train the algorithm. The more varied and voluminous the data is, the more areas it can cover. Consequently, with the right data, it can be used for a wide variety of use cases including business registration, dispute resolution, social security, open data and more. Six key application areas that eTranslation covers are displayed in figure 6. Consequently, it empowers public administrations, citizens, and businesses in the EU to use digital services in any of the 24 languages available in the tool. Important to note is that eTranslation is a full-scale NLP-as-a-service, rather than a merely internally used tool.

[90] https://joinup.ec.europa.eu/sites/default/files/document/2020-07/jrc120399_Misuraca-AI-Watch_Public-Services_30062020_DEF_0.pdf

[91] https://www.sciencedirect.com/science/article/abs/pii/S095070511830128X

[92] https://www.gov.uk/government/case-studies/natural-language-processing-for-land-registry-documentation-in-sweden

[93] https://nio.gov.si/nio/asset/semanticni+analizator+besedil

[94] https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eTranslation

[95] https://ec.europa.eu/inea/en/connecting-europe-facility

Figure 6: Key application areas of eTranslation

4.1.2. How was it developed?

The input data for training the eTranslation tools is a collection of resources that covers all languages of the 30 countries associated with the CEF Telecom programme. Furthermore, eTranslation started by building upon decades of work done by EU translators, enabling it to understand context-specific EU policy and legal terminology. For languages that have less resources, additional coverage (external databases) is provided so that the performance of the respective translation is guaranteed. Also, the data aims to cover as many domains as possible. For this purpose, domain-specific corpora, and terminology resources (e.g., lexica and dictionaries) in the areas of social security, consumer rights, legal domain, culture, health, public procurement are included in the training data[96]. This is key for the creation of domain specific engines within the eTranslation service. To effectively train the model, exceptionally large datasets are needed. More specifically, DGT started with approximately 500 000 sentence pairs (each pair consisting of a sentence in one language and its translation in another language) in a variety of languages to train the model. Having access to these amounts of data is not self-evident, and multiple ministries and central banks contributed to the provision of this data. It follows that building such extensive tools from scratch is a near impossible task if no other stakeholders are involved, simply because of the input data needed. This is why most NLP tools today will be built on previously made efforts in the field.

To prepare the data for this pipeline, DGT built a converter based on the open-source tool Okapi Tikal[97]. This tool allows for the conversion of different document formats to a uniform output format such as

[96] https://cef-at-service-catalogue.eu/catalogue/browse/78e1866a-6e46-4fa4-8032-dfcbae3367fa/

[97] https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eTranslation

XML, HTML, or others. Extensive text cleaning is then performed, including deletion of irrelevant data, modifications, and language specific treatments[98].

DG Translation started development of the tool in 2010, using Statistical machine translation (SMT) based solutions. This rather early method relied on statistical models to perform the translations, and soon appeared to be insufficient to reliably translate all EU languages. Therefore, it has made the transition towards state-of-the-art Neural Machine Translation (NMT) methods, resulting in a better performance of the tool[99]. Neural Machine Translation uses an artificial neural network to predict the probability of a sequence of words, usually modelling complete sentences in a single integrated model[100]. Put differently, each word in a sentence is represented by a number, resulting in a sequence of numbers. The neural network then predicts a resulting sequence of numbers, which represents the translated target sentence in another language. The prediction made by the algorithm is based on training data used to train the model, as described in the previous section on neural networks.

Using NMT has several key benefits and challenges in the context of eTranslation. An advantage of the Neural MT method is that complex sentence structures can be understood quite well. Moreover, incremental updates and adaptations can be facilitated by re-training the algorithm with new data. This improves the maintainability of the translation tool. However, NMT has some weaknesses as well. Since NMT's use large volumes of data to train, they can lack consistency. Thus, the same word might be translated slightly differently in different scenarios[101]. Finally, measuring MT quality can be highly complex, making it difficult to assess the accuracy of the model. The complexity of these models also implies that NMT structures often become a black box, making it nearly impossible to explain why the algorithm made certain choices.

Another important decision the DGT team - and any other team building an NLP application - needs to make is whether to locate the servers and infrastructure on premise or on the cloud. Both choices have their advantages, but in this case DGT decided to run their activities on premise. Being a European institution that deals with large amounts of (private) data, privacy and control is priority. Even though costs are lower when the infrastructure is cloud-based, the risk of giving away data to other parties is one they want to avoid at all costs. Furthermore, a higher degree of flexibility can be reached when working on the cloud, since additional computing power or storage space can be requested on demand. However, when data is kept on privately owned hardware, full control can be retained, and privacy can be guaranteed. This example shows that there are important decisions to be made when choosing how to organise servers, virtual machines, and data storage.

4.1.3. Who can use it?

eTranslation is free of charge and easy to adopt for a variety of user groups. A first important user group are European small and medium-sized enterprises. The eTranslation tool can enable these organisations to boost their international business activities in an automated and cheap way. Secondly, public service officials use eTranslation in their daily communication with other areas in Europe. A third user group can be public sector service providers and information system owners. By using an API, they can integrate

[98] https://wt-public.emm4u.eu/Resources//Documents/DGT-TM_EUR-LEX-preprocessing.pdf

[99] Eisele, A. (2019). (rep.). *eTranslation: Steps Towards Domain-Adaptive Neural MT*.

[100] https://en.wikipedia.org/wiki/Neural_machine_translation

[101] https://www.translatefx.com/blog/what-is-neural-machine-translation-engine-how-does-it-work?lang=en

eTranslation's machine translation capabilities into their digital services[102]. To use eTranslation, any user simply needs to create an EU Login account, create an eTranslation profile and choose the type of documents they wish to translate.

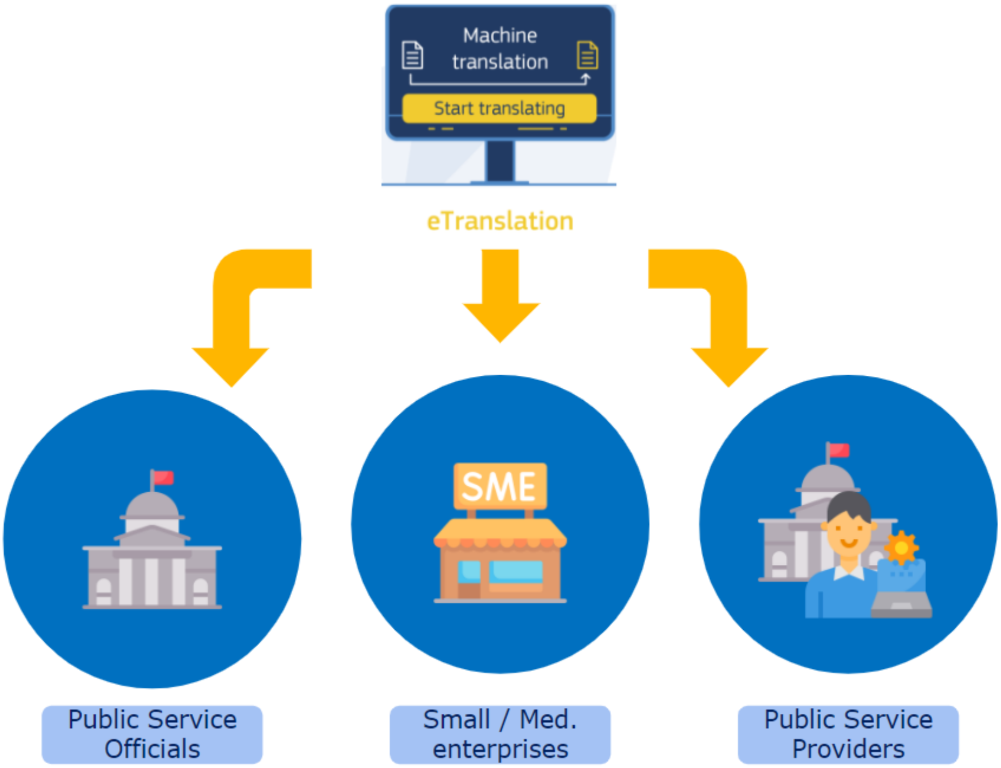


Figure 7: Main user groups of eTranslation

4.2. Example 2: NLP in education (DEAP and WollyEDU)

4.2.1. Context: DEAP

A second practical example of NLP used in the public sector was done in the context of Action 10 of the Digital Education Action Plan (2018-2020)[103]. At this time DG EAC (Education, Youth, Sport and Culture - European Commission) started cooperating with DG DIGIT (Informatics - European Commission) to identify possible pilot projects using AI and analytics in education. The cooperation started in January 2019 and, through only six months, allowed it to identify and develop two Proofs-of-Concept (PoCs)[104]. The PoCs developed in the framework of the cooperation between DG EAC, and DG DIGIT serve as a basis for Skills Gap and AI Impact (link to the tool can be found in the footnote at the bottom of this page).

Skills Gap and AI impact have been developed with two different purposes:

Skills gap links the skills acquired in specific educational programmes in Higher Education Institutions to those required by specific jobs in the labour market. Skills related to the job market and targeted courses in Higher Education were mapped using data from several universities from all around the EU.

AI Impact identifies what AI aspects or topics should be part of higher education AI programs to address current technological breakthroughs. It was developed by analysing articles within the most influential

[102] https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eTranslation
[103] https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=COM:2018:22:FIN
[104] https://d2eola03gl0y2i.cloudfront.net/screen/service1/exploreskills

scientific journals on AI breakthroughs and matching them with the topics covered in AI courses and programmes, delivered by front-running universities with a specific expertise on the topic.

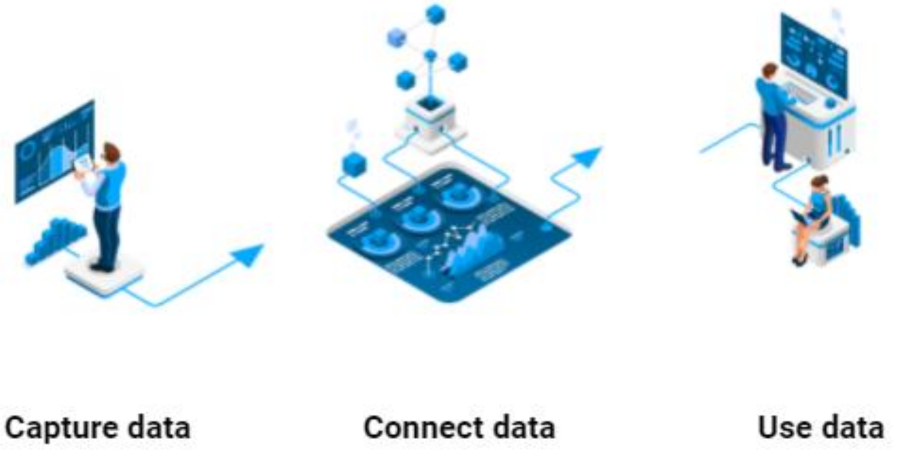


Figure 8: Data flow in DEAP

4.2.2. How was it developed: DEAP

In this project, there were two different data sources:

- Data from Higher Education Institutions describing their programmes.
- Description of skills and occupations from the ESCO database.

Each of these data had its own challenges. The data from Higher Education Institutions were completely unstructured since each University in Europe can provide curriculum to students in a different way. To overcome this challenge, a data template that Universities needed to fill with their data was designed. Furthermore, the data was written in multiple languages. For technical feasibility, the scope of the project was to only have English data processed.

The data from ESCO were of high quality with all official EU languages; however, the wording of the description of the skills were quite different from the one provided by Universities. It led to unexpected results in the matching later in the project.

After some desk research, it was agreed with all stakeholders to go with BERT since it is a state-of-the-art NLP model, and it was surpassing older models such as Word2Vec and Gloves on the NLP task used in this project: Semantic Text Matching.

AI transparency was also a huge topic since one requirement was to make Higher Education Institutions understand the process of the matching. Instead of using an AI transparency layer such as SHAP[105] or LIME[106], the stakeholders decided to display the similarity score to the user as well as both texts.

As the online services were intended to be available freely on the web, a container environment has been set up using the capabilities of the EC data platform. Of course, the whole cloud architecture was designed not only to deploy the NLP models but also to contain all cloud components necessary to run the public services from the storage of the data to the handling of the user traffic between the different containers.

[105] https://christophm.github.io/interpretable-ml-book/shap.html

[106] https://christophm.github.io/interpretable-ml-book/lime.html

4.2.3. Who can use it: DEAP

These services have been made available online, leveraging the infrastructure of the European Commission Data Platform, and are open for use through the internet without any restriction. The primary users will be stakeholders from the higher education sector, but also policy makers in DG EAC and other Directorates-General that potentially need data to define policy or research priorities on the links between the skills needed in the labour market and those provided by higher education institutions.

4.2.4. Context: WollyEDU

An example of NLP in education on a local level is WollyEDU[107]. WollyEDU is an online analysis system that allows the University of Padova (Italy) to gather information about the Italian Labour Market. This tool can compare the professional profiles coming out of the courses of the University of Padova with 6 million jobs advertised in Italy in the last 6 years. It highlights the most important elements of the labour market such as:

- the most requested occupations, down to the detail of each single industry.
- skills that distinguish each single occupation, which can be used as a tool to choose the degree courses.
- the time evolution of the number of job advertisements, to identify emerging skills or competencies.

4.2.5. How was it developed: WollyEDU

The tool was made by Burning Glass Technology using Named Entity Recognition based NLP tools. The developers encountered several key challenges in the development of the tool. Firstly, they needed to achieve a deeper granularity than ESCO Occupation[108] which is a classification made by the European Commission. This handbook lists all European skills, competences, qualifications, and occupations that are applicable to the European labour market. Next, the team aimed to match the completeness and terminology adopted by the documents describing the universities courses. This means they had to extract the relevant information from these documents, opposed to reusing existing data. The result was a tool that was tailored to the courses of the University of Padova and the Italian job market.

4.2.6. Who can use it: WollyEDU

The tool is used by the Career Service department of the university of Padova to give students a better understanding of the labour market. It allows these students to gain insights into the practical outcomes of the courses available, and to link these with career paths in the Italian labour market.

4.3. Example 3: MIREL

4.3.1. Context

The production of laws, courts' decisions in cases or, in general, legal texts is one of the key prerogatives of the public sector. Being able to reason with legal data has been a major research area for several years.

A practical example of a project that deployed NLP on public legal data is MIREL (MIning and REasoning with Legal texts). Funded by the EU, the aim of this initiative was to translate legal texts into formal representations that could be used for querying norms, compliance checking, and decision support.

[107] https://www.tabulaex.com/en/case-history/universita-di-padova/

[108] https://ec.europa.eu/esco/portal/document/en/0a89839c-098d-4e34-846c-54cbd5684d24

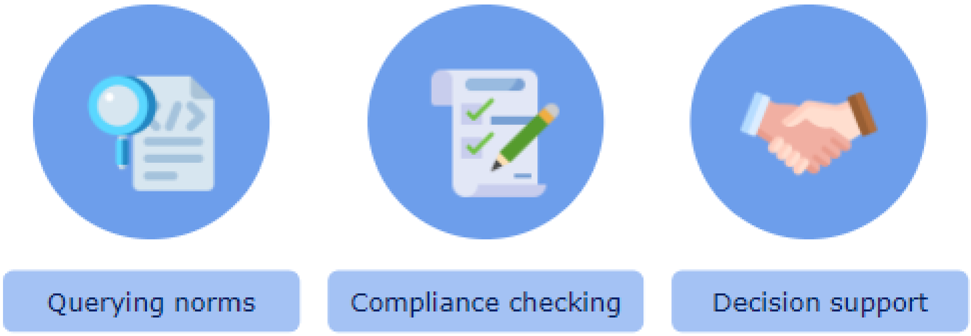
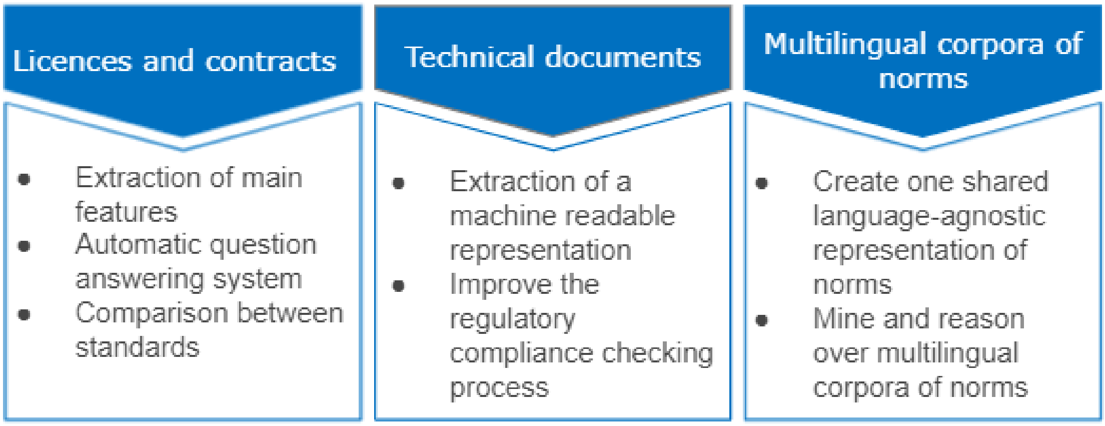


Figure 9: Use cases in MIREL

The initiative was an overarching collection of research and applications, with three envisioned usage scenarios including *licences and contracts*, *technical documents,* and *multilingual corpora of norms*.



Licences and contracts

- Extraction of main features
- Automatic question answering system
- Comparison between standards

Technical documents

- Extraction of a machine readable representation
- Improve the regulatory compliance checking process

Multilingual corpora of norms

- Create one shared language-agnostic representation of norms
- Mine and reason over multilingual corpora of norms

Figure 10: Mirel usage scenarios

The research project consisted of a consortium of 16 universities and research centres in 11 countries, closely collaborating with industrial stakeholders. Although the MIREL project is now finished (2017), it is still particularly relevant for two main reasons: on the one hand, it enabled the creation of an international and inter-sectoral network which defined a formal framework around NLP for legal data, and, on the other hand, it created tools and techniques for mining and reasoning with legal texts. For example, some of these NLP-techniques were combined with syntax- and logic-based extractions that detect dependencies between chunks of legal text. In other words, if applied to legislative texts, it could detect dependencies or overlaps between different laws, at any jurisdiction levels, to monitor at a granular level, the compliance between levels (e.g., national law to European law) or to ensure overall coherence when drafting a new law.

4.3.2. How was it developed?

MIREL had an especially large scope with respect to texts in many areas of EU regulations. Therefore, we will focus below on one of the many research questions tackled by MIREL: *how can NLP and ontologies help to classify and connect legal norms?* This is particularly disruptive considering that the knowledge of a legal corpus and all the cases that form the jurisprudence is today captured by humans. The ability to list all related cases would improve the quality of the judgements, make them more definitive and less dependent on the expertise of people.

Another example of a research question was h*ow can NLP and ontologies facilitate the monitoring of legal compliance?* MIREL investigated the example of GDPR and the complexity for personal data users and controllers to make sense of the various legal texts and verify their correct implementation.

MIREL studied the added value of ontologies for reasoning and processing legal data for several reasons that represent barriers for common NLP:

- The meaning of terms in the legal language often differs from the common language.
- In legal language, terms can have multiple meanings (i.e., what MIREL describes as the polysemy of terms). Ontologies and data models can provide formal definitions and relations between concepts to increase the ability of NLP tools to understand context, syntax, and semantics[109]
- Legal texts may oppose or contradict each other. Ontologies can help to relate legal norms to enable complex reasoning, for example in the case where two or more "*independent regulations are applied in the same circumstances*"[110].

By using ontologies, MIREL shows that NLP can be used for tagging, classifying, or translating (cfr. Example 1: eTranslation) legal texts.

As part of MIREL, ontology is defined as "*A formal* [machine-readable] *specification of a shared conceptualisation of a domain of interest*". For example, the Catalogue of Services Action has defined a vocabulary for describing public services (the Core Public Service Vocabulary Application Profile or CPSV-AP[111]). From this perspective, the main objective of ontologies is to propose a common understanding of a certain reality to information systems and people using it. In the legal domain, an example of an ontology for legal resources is ELI[112] (European Legislation Identifier). ELI defines a common data model for exchanging legislation data. The core of ELI is visualised below:

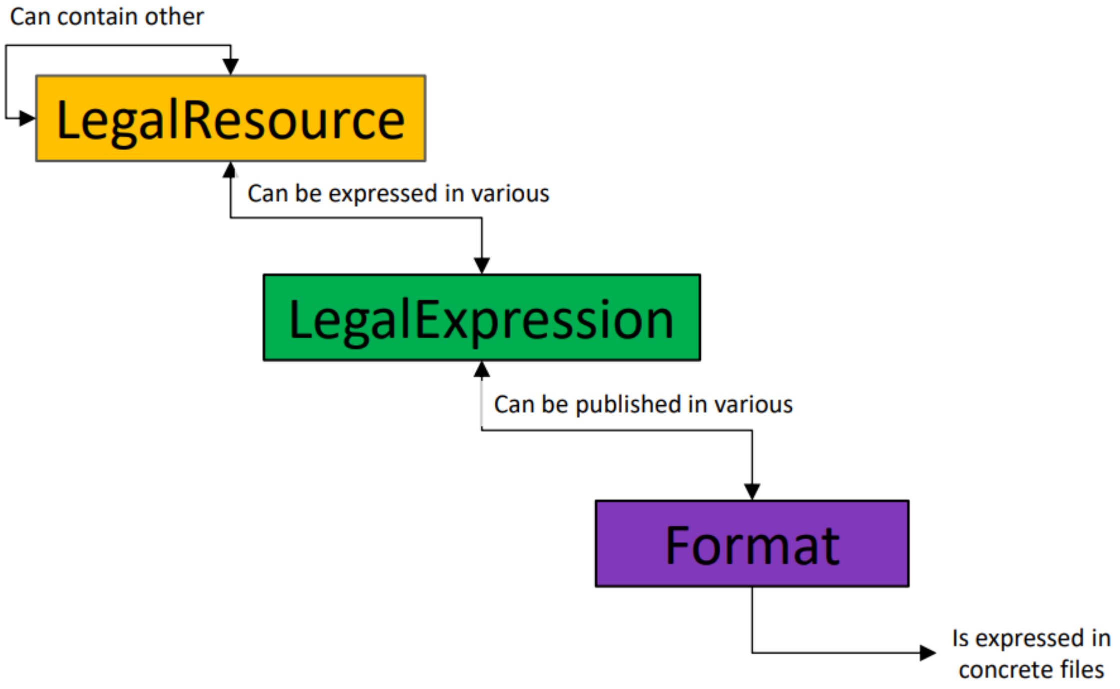

Can contain other
LegalResource
Can be expressed in various
LegalExpression
Can be published in various
Format
Is expressed in concrete files

Figure 11: structure of European Legislation Identifier (ELI)

[109] https://aclanthology.org/W04-0609.pdf

[110] Deliverable D3.1: Ontology-based access to normative knowledge (), In , 2017, p.6, accessible from: https://www.mirelproject.eu/publications.php.

[111] https://github.com/catalogue-of-services-isa/CPSV-AP/tree/master/releases/2.2.1

[112] https://op.europa.eu/en/web/eu-vocabularies/eli

The term ontology covers various specific types of specifications (e.g., taxonomy, vocabulary, code list) and formats (e.g., OWL, RDF), enabling different levels of reasoning with axioms or rules. As such, ontologies support the reasoning and processing of data (e.g., text, numbers, dates) by representing legal norms.

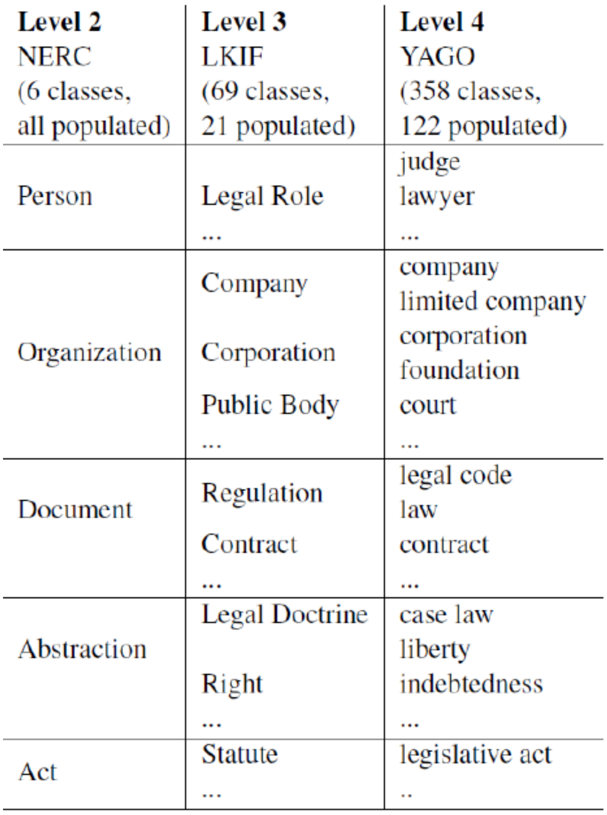
The first example from MIREL focuses on Named Entity Recognition and Classification (NERC). As mentioned in chapter 3.4.2, this technique aims at categorising words based on predefined categories called named entities. The ability to detect named entities and describe them precisely with hierarchies and sub-domains is crucial in the legal domain. They gave two examples of the same text with different levels of recognition:

> Example 1.1. The [Court]$_{\text{organization}}$ is not convinced by the reasoning of the [combined divisions of the Court of Cassation]$_{\text{organization}}$, because it was not indicated in the [judgment]$_{\text{abstraction}}$ that [Egitim-Sen]$_{\text{person}}$ had carried out [illegal activities]$_{\text{abstraction}}$ capable of undermining the unity of the [Republic of Turkey]$_{\text{organization}}$.
>
> Example 1.2. The [Court]$_{\text{European\_Court\_of\_Human\_Rights}}$ is not convinced by the reasoning of the [combined divisions of the Court of Cassation]$_{\text{Yargıtay Hukuk Genel Kurulu}}$, because it was not indicated in the [judgment]$_{\text{Court\_of\_Cassation's\_judgment\_of\_22\_May\_2005}}$ that [Egitim-Sen]$_{\text{Education\_and\_Science\_Workers\_Union\_(Turkey)}}$ had carried out [illegal activities]$_{0}$ capable of undermining the unity of the [Republic of Turkey]$_{\text{Turkey}}$.

This shows that each Named Entity may have several levels of specification. The Named Entities in these two examples can be described in ontologies. Such ontologies may also contain additional information about each Named Entity. MIREL describes many different ontologies in its research, such as:

- WordNet
- ELTS (European Legal Taxonomy Syllabus)
- PrOnto

As part of their work, they investigated how legal ontologies can be populated to support named entity recognition, classification and linking. More specifically, MIREL looked at how named entities can be tagged in texts with ontologies. They used two different ontologies, YAGO (Wikipedia and WordNet-based) and LKIF. By mapping the two ontologies, they created manually a hierarchy of entities:



Level 2 NERC (6 classes, all populated)	**Level 3** LKIF (69 classes, 21 populated)	**Level 4** YAGO (358 classes, 122 populated)
Person	Legal Role ...	judge lawyer ...
Organization	Company Corporation Public Body ...	company limited company corporation foundation court ...
Document	Regulation Contract ...	legal code law contract ...
Abstraction	Legal Doctrine Right ...	case law liberty indebtedness ...
Act	Statute ...	legislative act ..

Figure 12: Hierarchy of entities

Their objective was, by training and testing different NERC models on texts from Wikipedia and on legal texts from the European Court of Human Rights, to observe the level of accuracy at which their models could recognise the Named Entities according to the ontologies.

To analyse this, they compared the results of several NERC models trained on Wikipedia's corpus and tested first on Wikipedia's corpus, then on the legal texts from the European Court on Human Rights . The models were the following:

- Support Vector Machines (SVM)
- Curriculum Learning (CL)
- reversed Curriculum Learning (reCL)
- MultiLayer Perceptron (MLP)
- Stanford CRF Classifier model

On Wikipedia's data, the CL achieved a level of precision of 77% for the 122 classes from YAGO ontology. This increased to 84% for LKIF (21 classes) and 91% for the 6 classes of level 2.

As expected, on the legal texts from the European Court on Human Rights, the precision drops for all models with 16%, 35% and 64% for levels 4, 3 and 2 respectively. If the level of precision is low, MIREL indicates that it is compensated by the low cost of this general approach: the examples obtained from Wikipedia are open source therefore free to use, and the automated processing with the models proposed do not require a lot of resources (time and materials).

Overall, this example shows how ontologies can be used to leverage NLP in the case of classification: with different levels of granularity in the ontologies, the user can decide which level of classification the model should follow that optimises the quality of the results for a use case.

4.3.3. Who can use it?

The envisioned goal of MIREL is in the first place to create a network to define frameworks and develop tools for mining and reasoning with legal texts. This means that it tries to bridge the gap between communities and sectors to enable state-of-the-art technologies in the space of legal texts. It follows that MIREL is not just a single tool or application. However, it can be concluded that the research done within MIREL is most relevant for legal companies with interest in automation or software companies active in the legal domain. Because of the project, mobility and knowledge exchange is enabled between different parties to exploit the vast potential of NLP, computational ontologies, and logic & reasoning.

4.4. Example 4: Improving public services by mining citizen feedback[113]

4.4.1. Context

Taking into consideration preferences and feedback in public service decision-making is a prerequisite for successful governance[114]. This involves a robust understanding of how and why users are satisfied or dissatisfied. Public participation often makes administrative decisions less straightforward, and, in many contexts, it is difficult to generate enough input to include users actively in the process of decision-making.

In this example, researchers investigated how information technologies and NLP can contribute to create a deeper understanding of user preferences in publicly funded primary care (GP) services in England, leading to more effective decision-making for improving user satisfaction. A comparison was made between conventional methods to collect public feedback such as experiments, qualitative research, surveys, behavioural studies and automated NLP tools scanning user reviews. The researchers used topic modelling to automatically define 20 topics from many textual reviews, gaining insights on the most important concerns and thoughts among users of public health services. They come to the conclusion that qualitative studies are unfeasible for decision-making due to small sample sizes and high data collection costs, and that surveys are reliable but offer only a limited sub-sample of citizen experience. Lastly, behavioural insights fail to explain the reasons behind these particular behaviours. They show that the use of machine learning to systematically include citizen voices represents a significant improvement in political and public policy decision-making.

4.4.2. How was it developed?

The research uses a dataset of 208.287 online reviews from 7.700 publicly funded primary care practises in England. Furthermore, the written reviews were complimented by a 5-star Likert-scale that allows users to rate six aspects related to GP service experiences. The six aspects in the questionnaire were as follows:

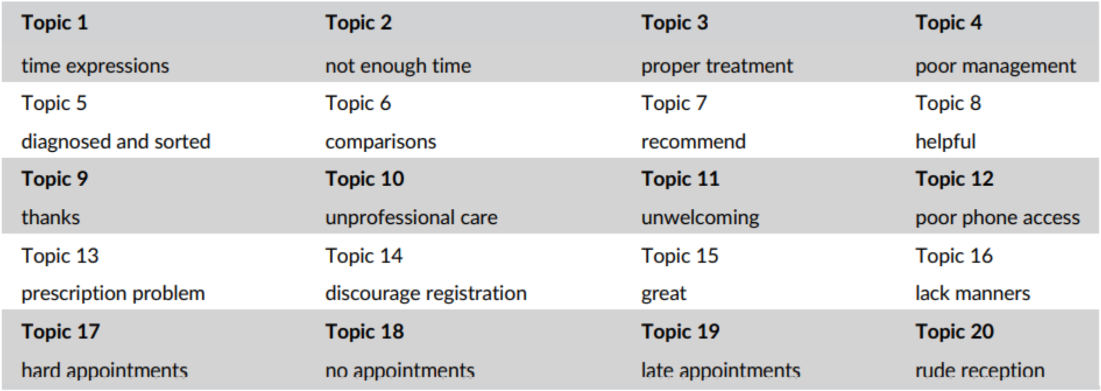
1. 'Are you able to get through to the surgery by telephone?'
2. 'Are you able to get an appointment when you want one?'
3. 'Do the staff treat you with dignity and respect?'
4. 'Does the surgery involve you in decisions about your care and treatment?'
5. 'How likely are you to recommend this GP surgery to friends and family if they needed similar care or treatment?'
6. 'This GP practice provides accurate and up-to-date information on services and opening hours'.

[113] https://onlinelibrary.wiley.com/doi/full/10.1111/padm.12656

[114] Feldman, D. L. (2014). Public value governance or real democracy. Public Administration Review, 74(4), 504–505

The written reviews were usually five or six sentences long and were pre-processed first. More specifically, all text was lower-cased and stemmed (i.e. similar to lemmatisation but with less complexity where a fixed portion of each word is 'cut off'), and numbers, punctuation and stop words were removed. Furthermore, each word with less than three characters or that appeared fewer than 10 times, or higher than 100.000 times, was removed from the corpus. The final corpus consisted of 9.148 unique terms that occurred over 8.5 million times in the dataset.

Next, topic modelling was applied to derive meaning from the reviews. Topic modelling is a highly suitable method because it can create near real-time summaries and insights from very large quantities of written texts. To do so, a human analyst was required to simply select the number of topics that the algorithm needed to uncover. The analysts in this example found that 20 topics was the optimal number for the objective, after which the model automatically identified the set of topics. Then, the proportional presence of each topic was approximated. The result is a set of 20 unique topics displayed in the table below:



Topic 1	**Topic 2**	**Topic 3**	**Topic 4**
time expressions	not enough time	proper treatment	poor management
Topic 5	Topic 6	Topic 7	Topic 8
diagnosed and sorted	comparisons	recommend	helpful
Topic 9	**Topic 10**	**Topic 11**	**Topic 12**
thanks	unprofessional care	unwelcoming	poor phone access
Topic 13	Topic 14	Topic 15	Topic 16
prescription problem	discourage registration	great	lack manners
Topic 17	**Topic 18**	**Topic 19**	**Topic 20**
hard appointments	no appointments	late appointments	rude reception

Figure 13: Result of the topic identification testing[115]

To further analyse the results, the researchers created a topic map that contains these topics. This representation shows comparison between the topics represented by the nodes and connectors. "*The greater the distance and the thinner the connecting line between two topics, the less likely they are to occur together within reviews. The node size of a topic corresponded to its popularity across patient reviews. Larger nodes indicated more common topics. Clusters of related topics are represented by node symbols. In this case, the ∨ symbol represented negative experiences, symbol ∧ topics represented positive experiences, and symbol = topics indicated themes without a strong positive or negative sentiment.*"[116]

The map is displayed in the figure below:

[115] https://onlinelibrary.wiley.com/doi/full/10.1111/padm.12656

[116] ibidem

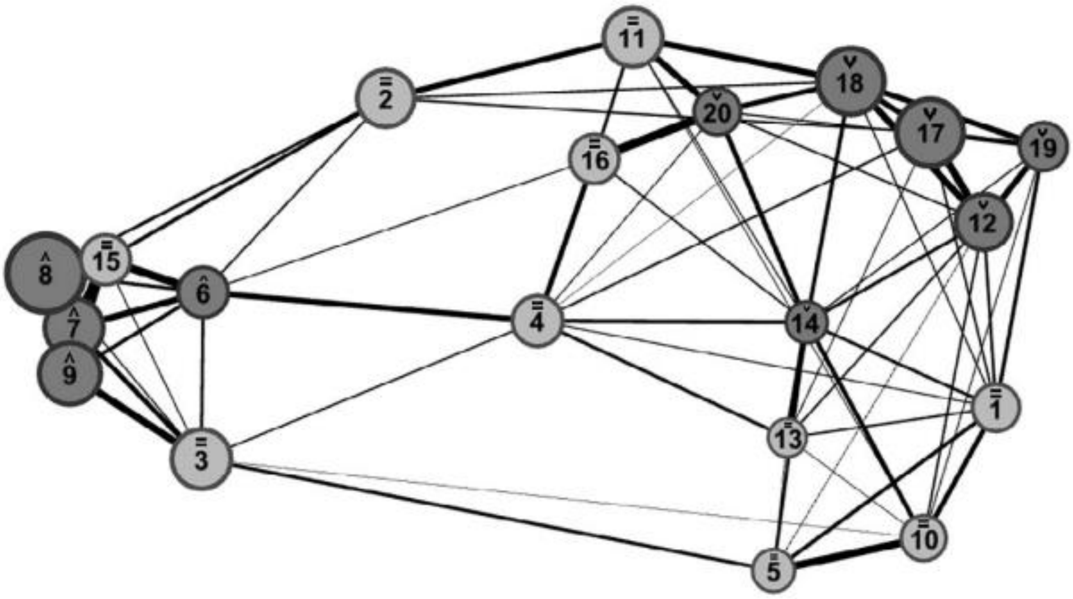


Figure 14: Topic map displaying the resulting 20 topics[117]

For instance, the "*lack of manners*" (16) and the "*rude reception*" (20) are very likely to be present in the same review. Similarly, topics *"hard appointments"* (17), *"no appointments"* (18) *"late appointments"* (19) and *"phone accessibility"* (12) are correlated strongly within the written reviews. It can also be seen that topics with very different contents are positioned far from each other, as is the case with *"helpful"* (8) and *"unprofessional care"* (10).

The researchers then proceeded to compare the outcome of the topic modelling to the results of the six Likert-scale ratings. Their findings suggest that the interactions with GP staff and reception staff are the two most important factors for predicting the satisfaction rating. The accessibility of GP services for making an appointment is the third biggest impacting factor on patient satisfaction. Improvements in this area could increase user satisfaction levels and decrease administration burden for staff. This gives the staff more opportunity to ensure proper interaction with patients, which can increase the quality of the user experience even further. The topic modelling exercise also revealed that "not enough time" (2) is a relevant concern of patients which is not included in NHS's GP Patient Survey[118]. It follows that topics that might be omitted in current methods of measuring user satisfaction will reveal themselves using automatic processing of unstructured feedback data.

4.4.3. Who can use it?

This example showcases how analysing user feedback can be achieved with topic modelling techniques in a scalable way to derive meaning for public decision makers and service providers in general. The topics described above, such as the ones causing the ease of appointment or the phone accessibility, can help to make decisions such as ensuring a 24h/7d phone accessibility or a Web-based booking app with centralised identification system for all services provided.

On top of this, the results of this analysis can be combined with data modelling techniques to improve the delivery of public services and future feedback analysis. More specifically, the robust topics identified

[117] https://onlinelibrary.wiley.com/doi/full/10.1111/padm.12656

[118] https://gp-patient.co.uk/surveysandreports

can be used in the context of public services to create a metadata model that structures and enriches semantically the relevant information resulting from the analysis. This way, public decision makers can make sure that the services described and provided answer these topics extracted from user feedback. For example, topic 12 (*poor phone access*) reveals the importance of having a well-integrated phone channel when dealing with public services. While in this research, the topics resulted from feedback received in the context of GP healthcare in England, such output could be relevant for any public service. In Europe, many administrations start from CPSV-AP[119] to describe their public services. CPSV-AP, the Core Public Service Vocabulary for cataloguing public services, includes generic attributes to describe a Channel as a medium through which people interact. In the Italian extension[120] of CPSV-AP, Channel was further specified to describe a Phone. More detailed needs from user feedback regarding the phone accessibility could be integrated in such a model to ensure that the way the data is structured or exchanged can be standardised across services. This would support the development of new processes to improve the phone accessibility or simply help to monitor in more detail the phone accessibility.

Similarly, topics 17 (*hard appointment*), 18 (*no appointment*) and 19 (*late appointment*) indicate the difficulty of arranging an appointment, which can be translated in the addition of clear communication regarding doctor occupation. In data models, a separate class or attribute containing this data could be defined, improving the specifications based on user feedback. By modelling information around appointments, a public administration could support the creation of a web-based tool or application for easily making an appointment and later on, logging the data input for further analysis. Decomposing the semantic (data model) and technical (tool, application) layers would help portability of the solutions created. For example, any public administration using the same data model could more easily adopt the tooling.

As ontologies can be used as a basis to structure information systems and how humans and machines can use it to process information, such improvements are relevant at service provider level or for the organisation responsible for the digitisation of the services.

Finally, modelling this kind of information, by extending existing elements such as Channel, Contact Point or Public Service attributes like processing time, would iteratively enable administrations to improve the way they capture, structure and analyse feedback.

[119] accessible from https://github.com/catalogue-of-services-isa/CPSV-AP/blob/master/releases/2.2.1/SC2015DI07446_D02.02_CPSV-AP-2.2.1_v1.00.pdf and developed under the former ISA2 (now Interoperable Europe) programme

[120] https://github.com/italia/daf-ontologie-vocabolari-controllati/blob/master/Ontologie/CPSV/CPSV-AP_IT.png

5. Proof-of-Concept: Automatic tagging of HTML pages

This chapter describes the steps and decisions taken to develop a proof-of-concept using NLP to automatically parse and tag web pages. This guides the reader through all the steps and decision points described in chapter 3 for a concrete case. Each step and decision made is justified to help the reader apply similar logic in his own NLP project. Moreover, all technical developments are open source and accessible from GitHub[121].

5.1. Statement of the issue

The Single Digital Gateway[122] is a regulation set by the European Commission in 2018 aiming to give access to citizens of Europe to Public Services falling into three categories: (i) Information, (ii) Procedure and (iii) Assistance services.

To facilitate the implementation of this regulation, the European Commission has created a repository of links where each Member States should insert the links or URLs of their web pages describing Public Services. These links will then be accessible by any citizen of the European Union via the YourEurope Portal[123].

All web pages referenced in the context of the Single Digital Gateway need to contain a set of metatags[124]. These metatags enable the European Commission to crawl the required information (through the EC Crawler). Examples of metatags required are the following:

Metatag name	Mandatory?	Description
sdg-tag	yes	The sdg-tag will contain the text sdg identifying the page as part of the Single Digital Gateway
DC. ISO3166	yes	It will contain the two characters ISO 3166-1 representation of names of countries.
DC. Location	only if applicable	Describes the NUTS or LAU location id for which the content on the page is valid (http://purl.org/dc/terms /Location)
DC. Service	yes	It will contain the information about the type of content present on the page (http://purl.org/dc /dcmitype/Service)
policycode	yes	It will contain the information about the code of the content area covered by the page according to Annex I and II or the full name of the assistance service in Annex III
DC. Policy	no, but recommended	It will contain the information about the name of the content area covered by the page according to Annex I and II (http://purl.org/dc/terms/Policy)

Table 4: Metatags in the Repository of Links

[121] https://github.com/catalogue-of-services-isa/SDG-PoC-on-Automatic-Tagging

[122] https://ec.europa.eu/growth/single-market/single-digital-gateway_en

[123] https://europa.eu/youreurope/index_en.htm

[124] https://ec.europa.eu/assets/grow/growth/_toolbox/sdg-docs/metadata-for-urls.pdf

Currently most web pages are tagged manually. This approach requires time and material for Member States.

To ease the tagging of web pages, the Catalogue of Services Action has decided, in the context of this deliverable, to build a proof-of-concept using Natural Language Processing. The aim of this proof-of-concept is twofold: (i) ease the tagging of web pages in the context of the Single Digital Gateway and (ii) provide a concrete example of what Natural Language Processing can bring to the public sector.

In practice, this proof-of-concept considers each metatag independently and gives the suitable value for each one of them. In this context, the development team defined a different strategy to annotate each metatag. This proof-of-concept and its method represent a first step for automating the tagging:

Metatag name	Strategy
sdg-tag	Gives the tag "sdg" regardless of the content of the web page. The goal of this tag is to indicate if the web page is referenced by the SDG and the YourEurope portal. In our case, all web pages need to be referenced.
DC. ISO3166	Gives the country present in the URL of the web page in the ISO3166 norm. For example, the web page http://grcontactpointcpr.ggb.gr/?lang=en will be considered as "GR" (Greece).
DC. Location	Gives the country present in the URL of the web page in the NUTS or LAU format. As an example, the web page http://grcontactpointcpr.ggb.gr/?lang=en will be considered as "ΕΛΛΑΔΑ" or "Greece".
DC. Service	Gives the DC.Service tag (Information or Procedure) using NLP to perform text classification. It has been decided to remove the Assistance Service tag (originally in the model) because: (i) this tag has never been found in the data given by the YourEurope portal. (ii) this tag has been removed in the updated version of the model[125]. As an example, the web page http://grcontactpointcpr.ggb.gr/?lang=en will be considered as "Information".
policycode	Gives the policycode tag using NLP to perform text classification. As an example, the web page http://grcontactpointcpr.ggb.gr/?lang=en will be classified as "M2" (product rules and requirements). Another method could have been chosen, Semantic Text Matching, to link the text of the web page with the description of the category. This option has been put aside because it was easier to perform Text Classification since it is the method used for the DC.Service and this method did not promise better results than Text Classification. As the development team had a short amount of time to perform the proof-of-concept, it has been decided to use Text Classification.
DC. Policy	Use the policycode tagged by the NLP classifier before to give its description. As an example, the web page http://grcontactpointcpr.ggb.gr/?lang=en will have the policycode tag "M2", therefore it will be tagged as "product rules and requirements".

Table 5: Metatags and their strategy for retrieval

As part of this proof-of-concept, the development team built an NLP engine for two metatags: (i) DC.Service and (ii) policycode. Furthermore, since English is the main language used as part of the Single Digital Gateway, the proof-of-concept was restricted to English web pages only.

[125] https://catalogue-of-services-isa.github.io/SDG-search-service-model/releases/v1.00/

In this context, the next section focuses on the text classification method for both metatags.

As the two NLP engines require a custom-made classification, the development team decided to create their own NLP pipeline. Furthermore, this prototype used the already existing environment (i.e., an AWS Virtual Machine) as the basis to build upon. Finally, an NLP model was fine-tuned in order to get better results.

5.2. Make the team

The development team was composed of two people that tackled two different sides of the proof-of-concept: the first focused on building the architecture and the API while the second aimed at creating the Natural Language Processing model. Therefore, in practice, the first developer took the role of Data Architect and Data Engineer while the second executed his tasks as a Machine Learning Engineer, Data Scientist and Data Analyst.

In addition to this core team, important insights from business people (Data Steward) were captured to guarantee that the proof-of-concept was answering the business needs.

As mentioned in chapter 3.2, some roles, such as Chief Data Officer, are only relevant for bigger projects. It is advised to start small with people having multiple roles.

5.3. Prepare the data

5.3.1. Collect

The data of this project comes from the YourEurope portal and was given as a spreadsheet with the following columns:

Column	Description	Example
Title	Title of the web page	“Κανόνες και απαιτήσεις που ισχύουν για τα δομικά προϊόντα”
URL	URL of the web page	“http://grcontactpointcpr.ggb.gr/”
Language	Language of the web page in the NUTS format (in this format “el” is Greek)	“el”
Classification information	Policycode of the web page	“M2”
Metadata type string	DC.Service code of the web page	“Information”
Country	Country of origin of the web page	“Greece”

Table 6: Input data overview

To use the data, the development team first needed to extract the text from each web page because this text was the main input necessary to build the NLP models. This task was tackled using the Python library BeautifulSoup[126]. This library was also used to extract the labels present in the web pages.

Overall, we had a corpus of 5215 web pages written in all languages of the EU and 2528 webpages in English. In practice, as we limited ourselves to English webpages, only the 2528 relevant web pages were used to train the language model as well as the classifiers.

5.3.2. Transform

Once all the web pages were scraped, the data was transformed in the right format to be processed. In this case, the intention was to perform text classification to categorise both the Policycode and the DC.Service tags. Therefore, the data was transformed in the following format:

Data	Description
Text	Text of the web page
DC.Service	DC.Service tag coming from the Metadata type string column of the YourEurope portal's dataset
Policycode	Policycode tag coming from the Classification information column of the YourEurope portal's dataset

Table 7: Data transformation format

5.3.3. Clean

At this point, the data was in the right format and needed to be cleaned. The cleaning step is particularly important since it will directly impact the performance of the NLP model. In our case, the following techniques were used:

- Remove spaces and redirects
- Remove stop words
- Lemmatise words
- Conversion to lowercase, strip and remove punctuation

These techniques were used because they were simple, and they were required data wise. Other techniques can be used such as automatic translation (some web pages labelled as English contained portions of texts in other languages). This kind of task would be particularly interesting to perform in a second phase of this project.

5.4. Create the NLP model

5.4.1. Word embedding

Once the data was cleaned, it was transformed into vectors using a word embedding algorithm. In this case, multiple word embedding could have been chosen but the development team decided to use Word2Vec (described in chapter 3.4.1). This choice was made for three reasons: (i) this word embedding is a well understood model, (ii) application of Word2Vec for text categorisation is well documented and (iii) this model is relatively simple therefore the development team did not need a specific infrastructure to train it (as opposed to BERT).

[126] https://www.crummy.com/software/BeautifulSoup/bs4/doc/

The next step for this step would be to compare the Word2Vec model that we trained against other equivalent models such as GloVe.

5.4.2. Application layer

Finally, a classifier was used to classify the text according to the relevant tag. For the DC.Service tag a random forest classifier was used. The type of classifier was used because: (i) it is a common classifier for text classification and (ii) a comparison had been made between Multinomial Bayes Algorithm and Random Forest: the latter gave better results. In practice, the development team used two classifiers: one to know whether the web page was an "Information" or not and one to know whether the web page was a "Procedure" or not.

The result for both classifiers can be found below:

	Precision	**recall**	**f1-score**
Information	0.91	0.79	0.84
Non-Information	0.98	0.98	0.98

Table 8: Results of the 'information' classifier

	Precision	**recall**	**f1-score**
Procedure	0.97	1.00	0.99
Non-Procedure	0.95	0.69	0.80

Table 9: Results of the 'procedure' classifier

In the tables above, we observe that the classifier shows good results on precision while they can be improved on the recall. It means that among all the web pages categorised as "Information" or "Procedure" by the classifiers, most of them were real "Information" or "Procedure". However, among all the real "Information" and "Procedure" web pages, only 0.79 and 0.69 were correctly categorised by the classifiers.

Even if these are promising results, this first proof-of-concept would benefit from a second phase of analysis to compare different language models and different classifiers to improve the results before proposing it to member states.

5.5. Deploy the NLP application

5.5.1. Architecture

All the components explained below were integrated in an AWS virtual machine. As explained before, this prototype did not use any off-the-shelf NLP components of AWS.

This proof-of-concept was designed with three main components in mind:

- The NLP engine, responsible for performing metadata analysis based on the NLP model, as previously described.

- An API[127], responsible for making the NLP engine available as-a-service.
- The NLP client, a library that is responsible for requesting metadata to the API and enriching a web page with the metadata retrieved from the API.

The sequence diagram below illustrates at high level the interaction between the components:

1. The NLP client contacts the API requesting the metadata to enrich the web page that should be crawled by the EC Crawler.
2. The API forwards the request to the NLP engine.
3. The NLP engine performs an analysis of the text of the web page, finds the metadata requested and returns the result to the API.
4. The API returns the requested metadata to the NLP client which, in turn, enriches the web page with such metadata.

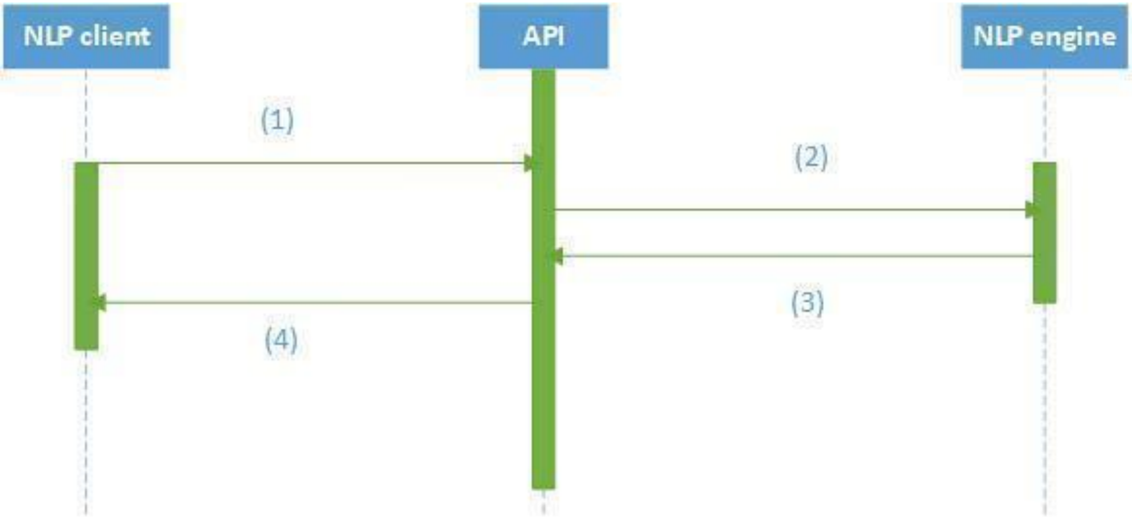


Figure 15: Sequence diagram for the Proof-of-concept

To develop these components, the AWS server available for the Catalogue of Service Action was used as depicted in the architecture below:

[127] https://publications.jrc.ec.europa.eu/repository/bitstream/JRC118082/jrc118082_api-landscape-standards.pdf

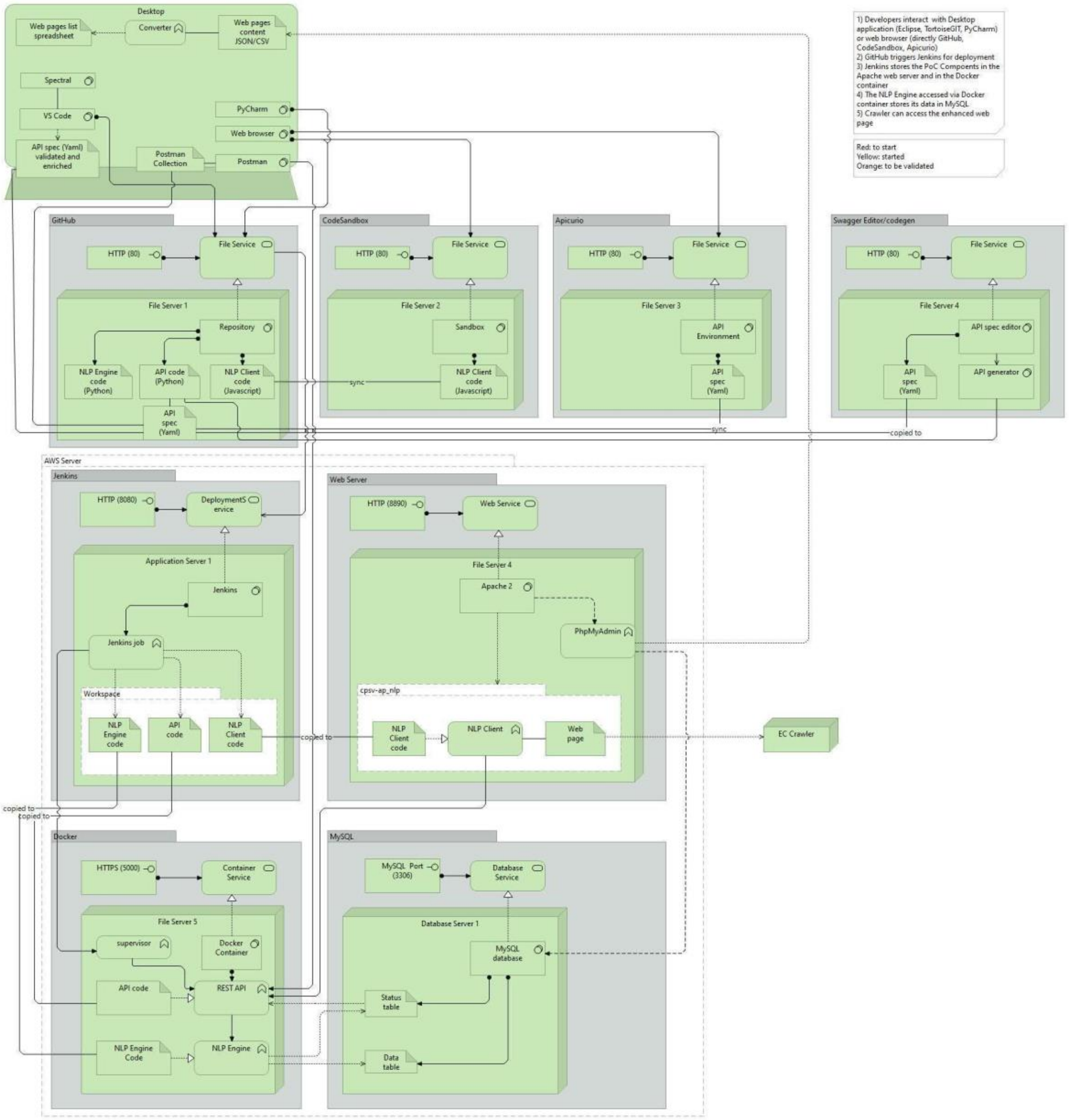


Figure 16: AWS server architecture

The architecture highlights 8 main blocks used for the development of the proof-of-concept:

1. GitHub[128], a service and repository of the code developed.
2. CodeSandbox[129], a service and development environment for the NLP client.
3. Apicurio[130], a service and environment to design the API and generate API contracts.
4. Swagger editor[131], a service and environment to edit an API contract and generate code from it.
5. Jenkins[132], as component setup on the AWS server, that automatically, for each new code on the GitHub repository, deploys the code of the NLP client, API and NLP engine on the AWS server.
6. Apache web server[133], as component setup on the AWS server, that allow to:
    a. Access a test web page for the API

[128] https://github.com/catalogue-of-services-isa/SDG-PoC-on-Automatic-Tagging-

[129] https://codesandbox.io/

[130] https://www.apicur.io/

[131] https://editor.swagger.io/

[132] https://www.jenkins.io/

[133] https://httpd.apache.org/

b. Access easily administrates the database, via the phpMyAdmin[134] component

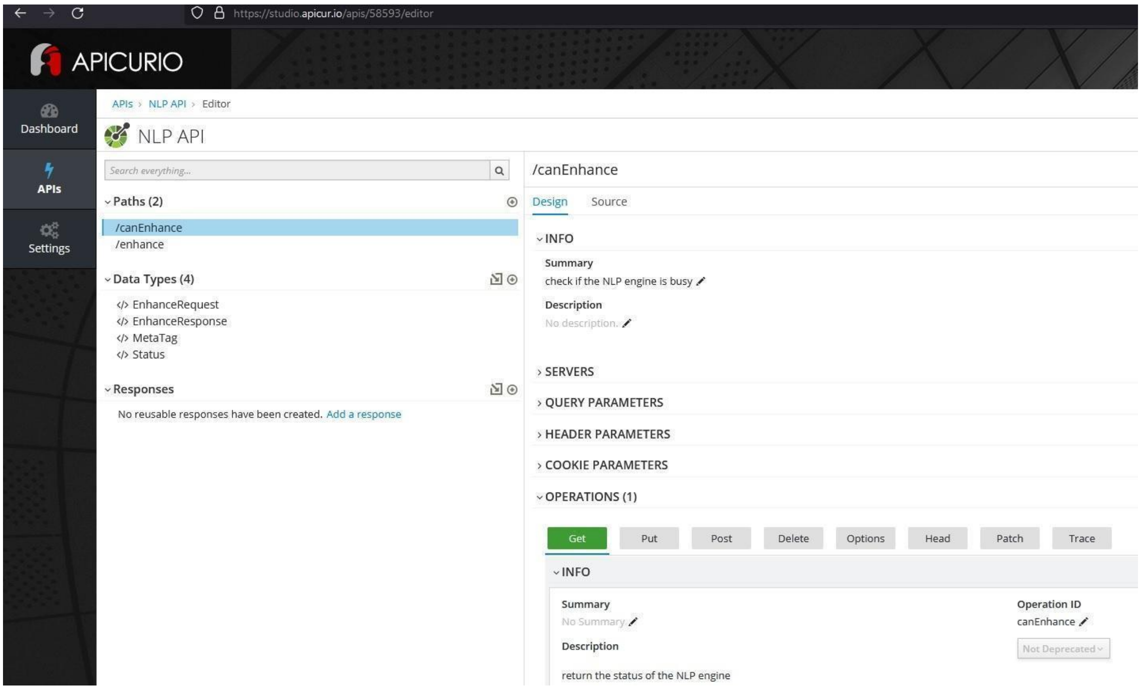
7. Docker[135], as component setup on the AWS server, acting as a container manager for the API and NLP engine.
8. MariaDB[136], as component setup on the AWS server, that allows storing the data for analysis of the NLP engine.

The development of the proof-of-concept started with the design of the API which is crucial for the NLP client and NLP engine.

5.5.2. API

The API has been designed as a Json REST API compliant with OpenAPI 3.0 specification[137].

The online service Apicurio has been used to design the API via a web form, generate the API contract and store it directly on the GitHub repository.



Summary
check if the NLP engine is busy

Description
No description.

Summary
No Summary

Operation ID
canEnhance

Description
return the status of the NLP engine

Figure 17: The Apicurio interface

The API has been designed with 2 operations named:

1. canEnhance; an HTTP GET operation to request to the database if the NLP engine is available to perform a metadata enrichment

   The result of the operation is a boolean (true or false) indicating “true” if the NLP is available:

```
{
 "status": true
}
```

[134] https://www.phpmyadmin.net/

[135] https://docs.docker.com/engine/

[136] https://mariadb.org/

[137] https://swagger.io/specification/

2. Enhance; an HTTP POST operation that sends the requested metadata to the NLP engine and gets back the metadata requested.

   The HTTP POST operation has as input the following structure:

   ```
   {
     "metatags": [
       "DC.ISO3166”
     ],
     "text": "Test",
     "url": "http://www.mywebsite.com"
   }
   ```

   Where:

   - metatags: it is the list of the metatags required by the NLP client to enrich the web page with metatags that will be parsed by the EC Crawler.
   - text: it is the text extracted from the web page where the NLP client is executed.
   - url: it is the URL of the web page where the NLP client is executed.

   The POST operation has as output the following structure:

   ```
   {
    "metatags": [
     {
      "name": "DC.ISO3166",
      "value": "IT"
     }
    ]
   }
   ```

   Where in the list of metatags, for each requested metatag there is its name and the returned value.

Once the design of the API has been concluded, the API contract has been:

- validated for conformance to best practices via the open source Spectral[138] tool which is a tool developed by the “Dipartimento per la trasformazione digitale” in Italy in supporting Agid when developing guidelines for the technical interoperability of public administrations[139].

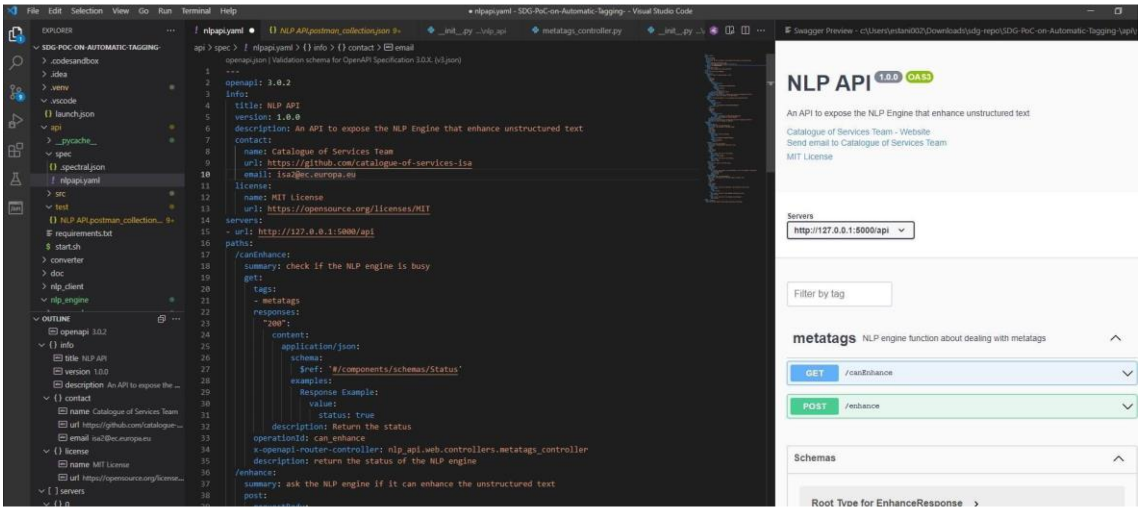
[138] https://stoplight.io/open-source/spectral/

[139] https://medium.com/@Developers_Italia/openapi-checker-il-verificatore-delle-interfacce-digitali-api-1d50b978c8c5

- enriched on the laptop of the developers, as the intention has been to develop the API in Python by means of the Flask framework[140], an open-source framework for API development, which requires, via the library Connexion[141], some specific configuration in the contract.

The code of the API has been developed via the free editor VS code[142] which has plugins:

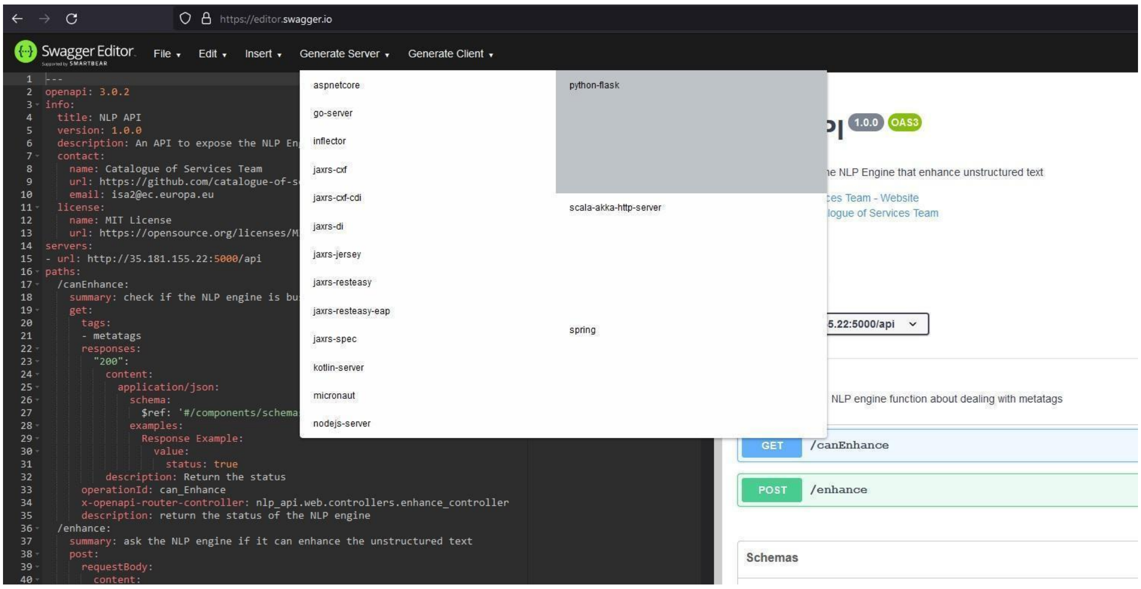
- To integrate with the GitHub repository.
- To integrate with Spectral for real time conformance.
- To preview the API operations.
- to run the API locally to test it.


nlpapi.yaml - SDG-PoC-on-Automatic-Tagging- - Visual Studio Code
EXPLORER
SDG-POC-ON-AUTOMATIC-TAGGING-
nlpapi.yaml
NLP API.postman_collection.json
metatags_controller.py
api > spec > nlpapi.yaml > {} info > {} contact > email
NLP API
1.0.0
OAS3
An API to expose the NLP Engine that enhance unstructured text
Catalogue of Services Team - Website
Send email to Catalogue of Services Team
MIT License
Servers
http://127.0.0.1:5000/api
Filter by tag
metatags NLP engine function about dealing with metatags
GET /canEnhance
POST /enhance
Schemas
Root Type for EnhanceResponse

```yaml
---
openapi: 3.0.2
info:
  title: NLP API
  version: 1.0.0
  description: An API to expose the NLP Engine that enhance unstructured text
  contact:
    name: Catalogue of Services Team
    url: https://github.com/catalogue-of-services-isa
    email: isa2@ec.europa.eu
  license:
    name: MIT License
    url: https://opensource.org/licenses/MIT
servers:
- url: http://127.0.0.1:5000/api
paths:
  /canEnhance:
    summary: check if the NLP engine is busy
    get:
      tags:
      - metatags
      responses:
        "200":
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Status'
              examples:
                Response Example:
                  value:
                    status: true
          description: Return the status
      operationId: can_enhance
      x-openapi-router-controller: nlp_api.web.controllers.metatags_controller
      description: return the status of the NLP engine
  /enhance:
    summary: ask the NLP engine if it can enhance the unstructured text
    post:
```

Figure 18: The API code part 1

Once the API contract is ready, the initial code has been generated via the service Swagger editor, which allows the browser to preview once again the API and to generate the initial Python backend code of the API.



```yaml
---
openapi: 3.0.2
info:
  title: NLP API
  version: 1.0.0
  description: An API to expose the NLP En
  contact:
    name: Catalogue of Services Team
    url: https://github.com/catalogue-of-s
    email: isa2@ec.europa.eu
  license:
    name: MIT License
    url: https://opensource.org/licenses/M
servers:
- url: http://35.181.155.22:5000/api
paths:
  /canEnhance:
    summary: check if the NLP engine is bu
    get:
      tags:
      - metatags
      responses:
        "200":
          content:
            application/json:
              schema:
                $ref: '#/components/schema
              examples:
                Response Example:
                  value:
                    status: true
          description: Return the status
      operationId: can_Enhance
      x-openapi-router-controller: nlp_api.web.controllers.enhance_controller
      description: return the status of the NLP engine
  /enhance:
    summary: ask the NLP engine if it can enhance the unstructured text
    post:
      requestBody:
        content:
```

Figure 19: The API code part 2

[140] https://flask.palletsprojects.com/en/2.0.x/

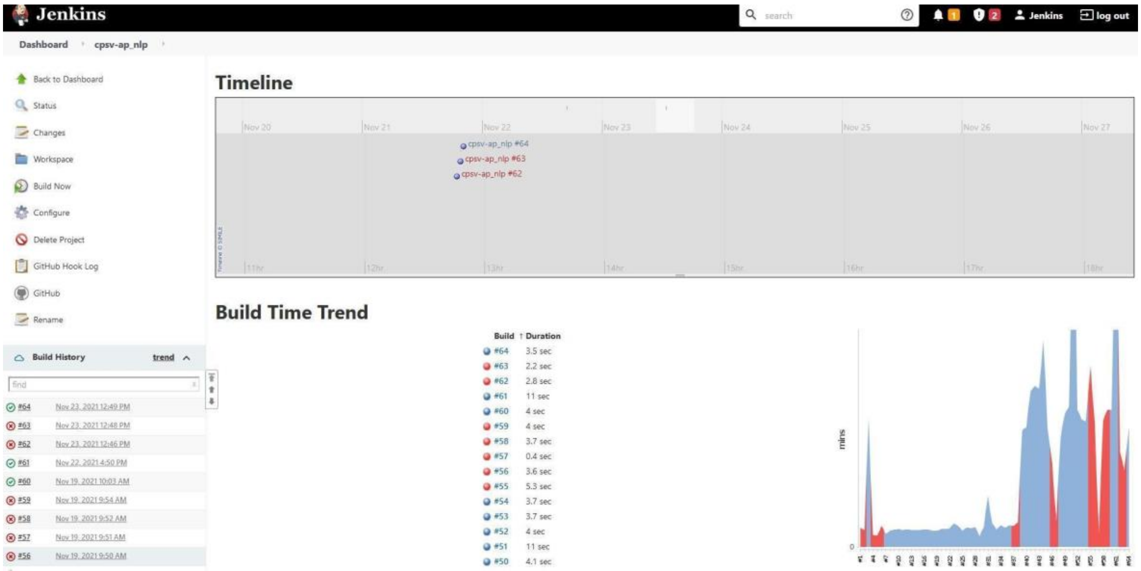
[141] https://connexion.readthedocs.io/en/latest/

[142] https://code.visualstudio.com/

The Python backend code has been further developed to execute the code of the NLP engine.

The API has been then automatically deployed into the AWS server by means of Jenkins which has also the scope to restart the API (hot-deploy) whenever there is a change in the code by using an open-source utility called Supervisor, depicted in the architecture diagram.

Jenkins provides a web interface allowing users to monitor the status of the deployment (build) to see how long it took and if it succeeded or failed.



Build History trend

✓ #64	Nov 23, 2021 12:49 PM
☒ #63	Nov 23, 2021 12:48 PM
☒ #62	Nov 23, 2021 12:46 PM
✓ #61	Nov 22, 2021 4:50 PM
✓ #60	Nov 19, 2021 10:03 AM
☒ #59	Nov 19, 2021 9:54 AM
☒ #58	Nov 19, 2021 9:52 AM
☒ #57	Nov 19, 2021 9:51 AM
☒ #56	Nov 19, 2021 9:50 AM

Build	↑ Duration
#64	3.5 sec
#63	2.2 sec
#62	2.8 sec
#61	11 sec
#60	4 sec
#59	4 sec
#58	3.7 sec
#57	0.4 sec
#56	3.6 sec
#55	5.3 sec
#54	3.7 sec
#53	3.7 sec
#52	4 sec
#51	11 sec
#50	4.1 sec

Figure 20: Jenkins web interface

Jenkins deploys the API and NLP engine code inside the container setup on the AWS server. The container has been created from an image pulled from the Docker Hub[143], a public registry of images; in the case of the proof-of-concept the official Python 3.9[144] image was used.

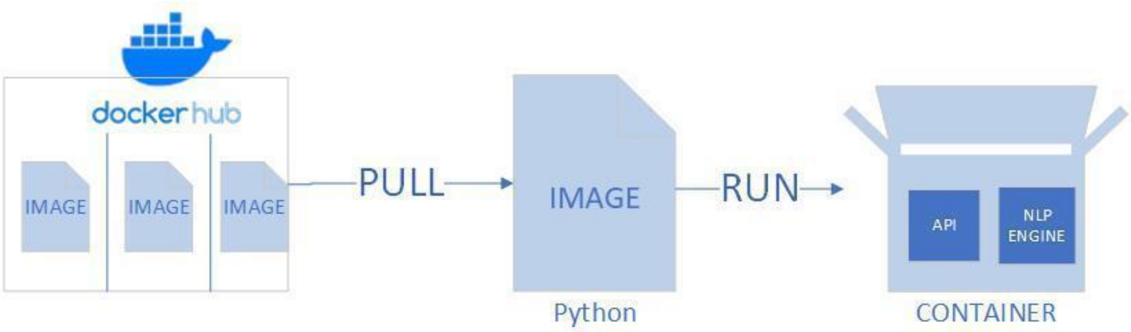


Figure 21: Container setup

[143] https://hub.docker.com/

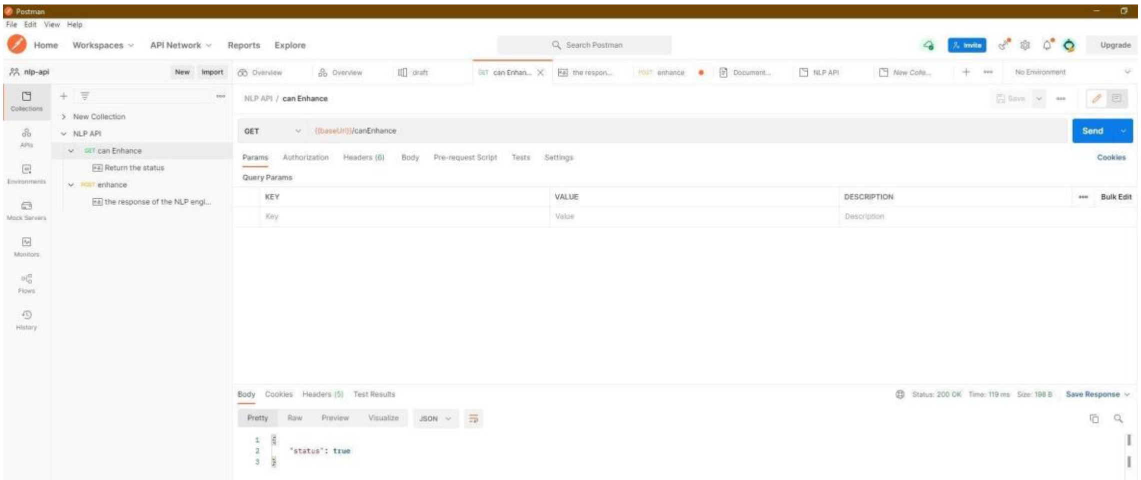
[144] https://hub.docker.com/_/python

Once Jenkins finished the deployment, the API operations could be reached at the following URLs (currently restricted to the development team):

- https://35.181.155.22:5000/api/canEnhance
- https://35.181.155.22:5000/api/enhance

Notice the HTTPS protocol in the URLs, indeed a self-signed certificate was installed allowing API to be reached, via secure connection, by the NLP client that was running in CodeSandbox on a secure protocol as well. It is worth noting that many of the MS portals using the SDG service rely on the HTTPS protocol, thus implementing the API under the HTTPS protocol is needed. In a real use case, an API gateway or an approved certificate should be put in place.

The API has been manually tested via Postman[145], a free application that can contact the API previously deployed starting from the API contract. Postman then can execute the two API operations and inspect the API response.



GET {{baseUrl}}/canEnhance

Status: 200 OK Time: 119 ms Size: 198 B

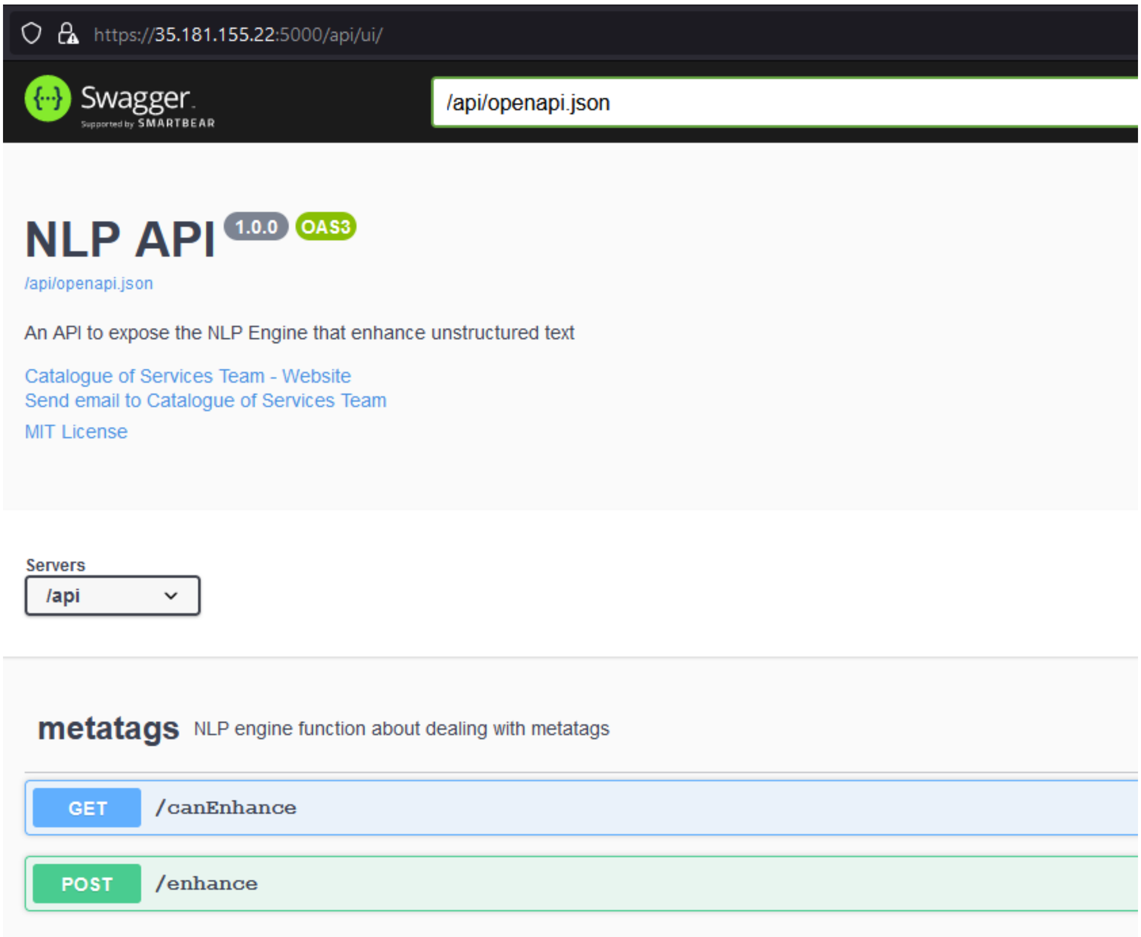
```
{
    "status": true
}
```

Figure 22: Testing the API via Postman

Furthermore, end users can access the interactive documentation of the API at:

https://35.181.155.22:5000/api/ui/

[145] https://www.postman.com/downloads/



https://35.181.155.22:5000/api/ui/

/api/openapi.json

NLP API 1.0.0 OAS3

/api/openapi.json

An API to expose the NLP Engine that enhance unstructured text

Catalogue of Services Team - Website
Send email to Catalogue of Services Team
MIT License

Servers: /api

metatags NLP engine function about dealing with metatags

GET /canEnhance

POST /enhance

Figure 23: The interactive documentation of the API

5.5.3. NLP client

The scope of the NLP client is to enrich HTML pages; thus, it has been naturally thought to be developed in JavaScript; in particular, it is based on jQuery[146], an open-source JavaScript library which allows an easy manipulation of web pages and performs operations with the REST API.

A manual analysis of the websites using the SDG service has been done to see which jQuery version was the most used; such analysis concluded to use jQuery v3.5.1 within the proof-of-concept.

[146] https://jquery.com/

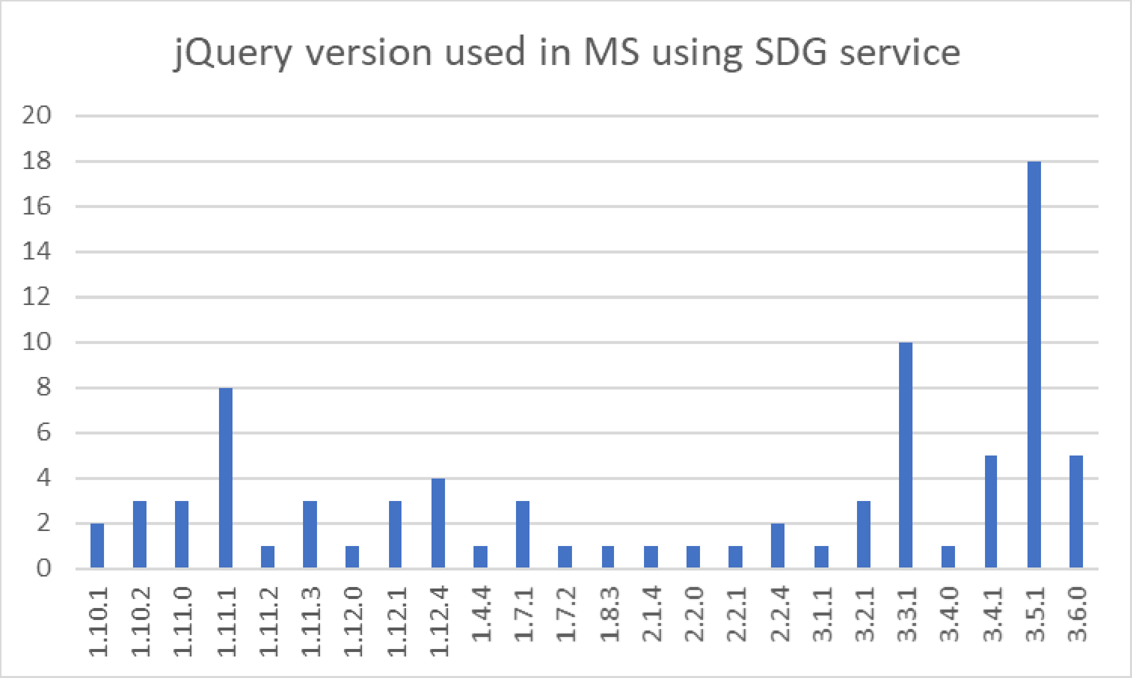
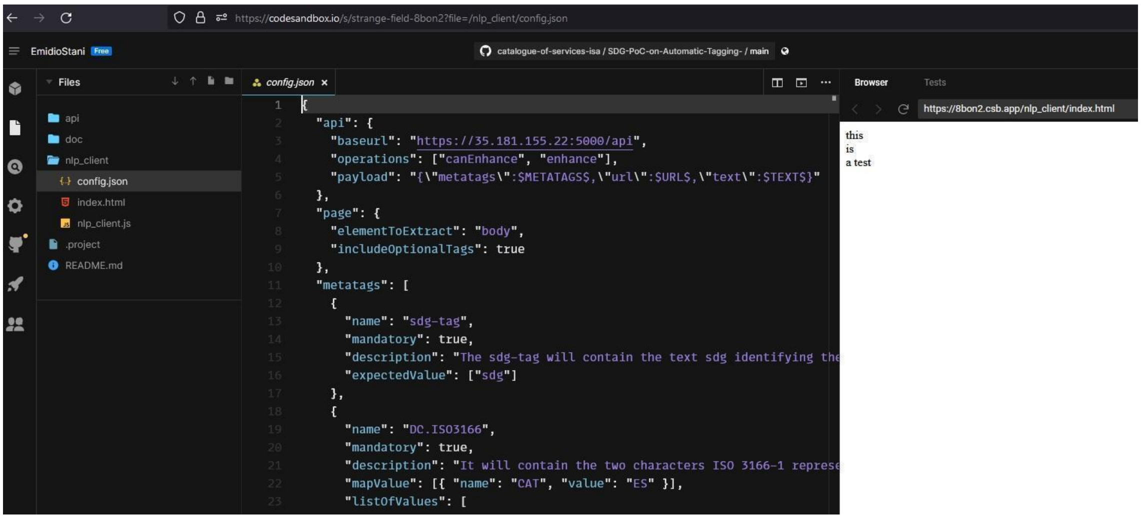


Figure 24: The most used jQuery versions

The NLP client has been developed by means of the CodeSandbox service which allows:

1. To edit the NLP client code providing coding assistance to developers.
2. To store on GitHub repository the code developed.
3. To test in real time the code developed.


https://codesandbox.io/s/strange-field-8bon2?file=/nlp_client/config.json
EmidioStani
Free
catalogue-of-services-isa / SDG-PoC-on-Automatic-Tagging- / main
Files
api
doc
nlp_client
config.json
index.html
nlp_client.js
.project
README.md
config.json
Browser
Tests
https://8bon2.csb.app/nlp_client/index.html
this
is
a test

```
{
  "api": {
    "baseurl": "https://35.181.155.22:5000/api",
    "operations": ["canEnhance", "enhance"],
    "payload": "{\"metatags\":$METATAGS$,\"url\":$URL$,\"text\":$TEXT$}"
  },
  "page": {
    "elementToExtract": "body",
    "includeOptionalTags": true
  },
  "metatags": [
    {
      "name": "sdg-tag",
      "mandatory": true,
      "description": "The sdg-tag will contain the text sdg identifying the
      "expectedValue": ["sdg"]
    },
    {
      "name": "DC.ISO3166",
      "mandatory": true,
      "description": "It will contain the two characters ISO 3166-1 represe
      "mapValue": [{ "name": "CAT", "value": "ES" }],
      "listOfValues": [
```

Figure 25: Configuration file of the NLP client

The NLP client first checks if metatags are present in the web page; as previously described , some are mandatory, others are optional. The missing mandatory metatags will be requested to the API, the optional metatags might be requested depending on the NLP client configuration, then the following actions happen:

1. The NLP client performs the first API request, canEnhance, to check the status of the NLP engine being available.
2. The API contacts the database to retrieve the status.
3. The database replies with the status to the API.

4. The API replies to the NLP client with the status.
5. In case of positive result, the NLP client performs a second API request, enhanced, to request the metatags (in case of negative response the NLP client stops).
6. The API contacts the NLP engine to perform the analysis of the text of the web page.
7. The NLP engine replies with metatags found to the API.
8. The API replies to the NLP client with the metatags found and the NLP client adds them to the page.

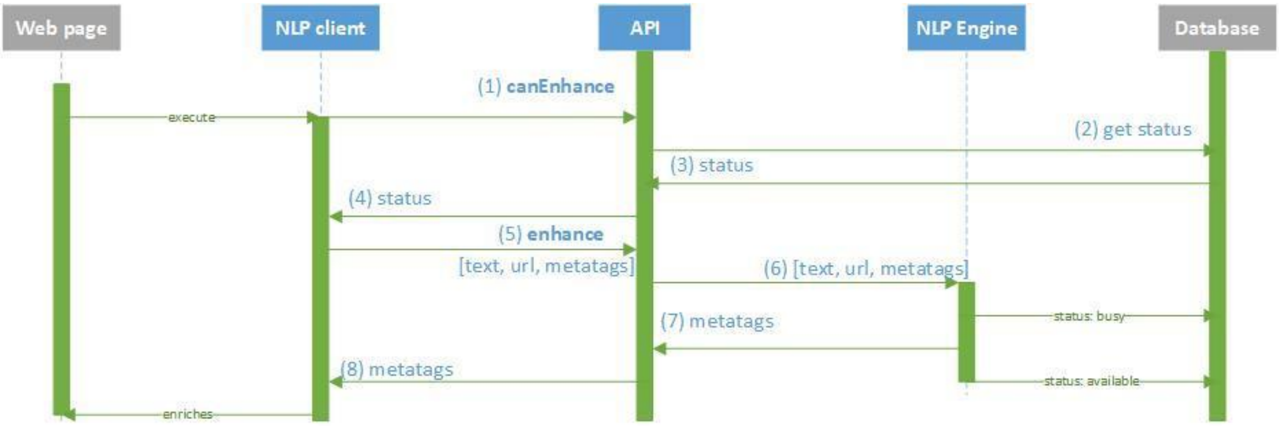


Figure 26: Sequence diagram displaying the client, API, database and engine

The NLP client uses a configuration file:

- To know how to contact the API (address and operations with their method, timeout, and message to be sent).
- To know how to manipulate the web page (which part of the page needs to be extracted and which metatags should be requested).
- To know which metatags to request (if they are mandatory and expected values).

Once the NLP client has been implemented, Jenkins has been configured to automatically deploy the NLP client and a sample web page on the AWS server that can be used for testing.

6. Conclusion

This report started by exploring several key application areas and possibilities of Natural Language Processing models. The application areas indicate how and in which context NLP can be used by organisations. This introduction provided the reader with a clear view and initial understanding of NLP in practice. Next, a hands-on guide on how to start building an NLP model was provided from identifying the data and building the team, to maintaining the model. Within these steps, the study also dove deeper into the language models that perform the embedding of the text, as well as the application layer creating the algorithm. To further investigate the potential of NLP, four concrete use cases of past or ongoing projects in the public sector were analysed. The use cases displayed in a practical manner how NLP can have a significant impact when the necessary efforts are made, potentially relieving many hours of manual text processing tasks or bringing new insights and consequently increased quality of services or products. The application areas in the public sector that surfaced from the examples were translation, education, text mining & reasoning and analysis of user reviews. Some key conclusions can be drawn from both the guide on building the model and the examples:

1. Public administrations and government agencies are exposed to highly varied and voluminous data. Careful data-preparation (collecting, transforming, cleaning) is imperative.
2. When creating an NLP model, a public organisation needs to consider the objective of the application, the complexity of the task, availability of out-of-the-box applications and availability of data that can be built upon.
3. NLP can leverage the use of standardised metadata and structured data models or ontologies, in domains where the semantic is complex such as with legal language.
4. NLP can be used to improve public service accessibility and effectiveness by:
    a. extracting useful insights from large amounts of data; and
    b. automating procedural and repetitive tasks.

Finally, this study describes a proof-of-concept on the automatic tagging of HTML pages. This proof-of-concept acts as a detailed fifth example giving practical details on the different steps followed to build, test, and deploy an NLP model.

Originally, this proof-of-concept was developed for two reasons: (i) ease the tagging of web pages in the context of the Single Digital Gateway and (ii) provide a concrete example of what Natural Language Processing can bring to the public sector.

So far, this proof-of-concept has succeeded on both its objectives. First, this project is an attempt to automatise the burdensome task of tagging web pages. As the results show, a lot of the work can be automated using this kind of tool. With more ambition, the impact of the YourEurope portal on the accessibility of Public Services can be important. Second, this project is a hands-on example on leveraging NLP in the public sector. As with many similar projects, the development team has been confronted with common challenges for this type of project: multilingualism, legacy IT components. This example can be used as a blueprint for other NLP projects in the public sector.

The Catalogue of Services Action may investigate several aspects for building further on top of the proof-of-concept:

- In a technical way, other strategies to produce the tags in the context of the Single Digital Gateway can be investigated and compared to achieve better results. Furthermore, other language models as well as other classifiers can be investigated and compared to obtain higher quality outcomes.

- In a business way, the automatic tagging can be generalised to the latest data model of the SDG[147] where more information is requested from the member states.

Finally, we would like to invite the reader to share concrete applications of NLP with the research team if applicable. If any projects have been conducted in your organisation that could be of interest in relation to this study, we would be highly interested in learning more about them.

These use cases can be shared on the following email address: DIGIT-INTEROPERABILITY@ec.europa.eu

[147] https://catalogue-of-services-isa.github.io/SDG-search-service-model/releases/v1.00/

7. Annex

7.1. Supervised, unsupervised and reinforcement machine learning models

Supervised, unsupervised and reinforcement machine learning models

Artificial Intelligence and Machine Learning algorithms have three different approaches of learning (training) a model: supervised, unsupervised learning[148] and reinforcement learning.

Supervised learning is when a machine learning algorithm uses a labelled dataset for training the model. This means that predefined categories or classes are used by the model to predict to which of these categories new data instances belong. Such an approach is necessary if the model requires business knowledge to perform its task. For example, a classification algorithm is trained to distinguish apples from oranges, or a regression model tries to understand the relationship between independent and dependent variables.

Unsupervised learning is when a machine learning algorithm analyses unlabelled data sets to cluster, associate or reduce them. These models aim to discover hidden patterns in datasets without humans having to intervene. For example, a market basket analysis where products can be related to each other when they are being bought by the same customer is an application area of unsupervised learning.

Reinforcement learning uses trial and error to come up with a solution to a problem like in a game-like situation. In practice, the artificial intelligence gets either reward or penalties for the action it performs. The goal is to maximise the total reward. As in unsupervised learning, one does not need labelled data to use this kind of paradigm. It is up to the machine to figure out how to solve the problem. For example, deepsense.ai[149] was able to train a virtual runner from scratch without giving him hints on how to do it.

7.2. Text classification techniques

The content of this section is technical and provides technical insights to build the Language Model

There are several text classification techniques as mentioned in section 3.4.2. Several of the most used examples are listed below.

Multinomial Naive Bayes algorithm

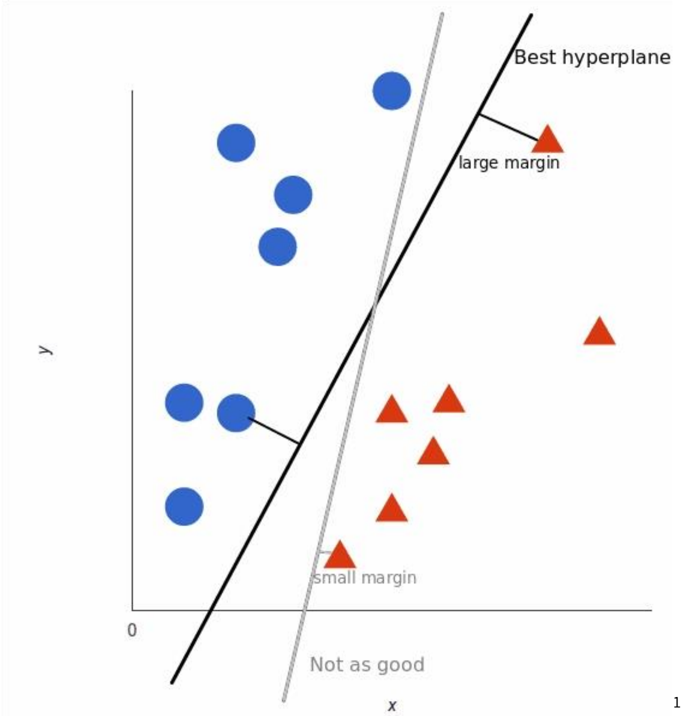
Many techniques exist to classify texts using machine learning. An example of a widely used and general technique is a multinomial Naive Bayes algorithm. The model works on Bayes theorem of probability to predict the class of unknown data sets. It is a universal technique for classification that is also used outside of NLP.

[148] https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning

[149] https://deepsense.ai/learning-to-run-an-example-of-reinforcement-learning/

Support Vector Machines (SVM)

More relevant to our purpose is another example of a text classification method called Support Vector Machines. This algorithm makes use of vectorial representations of text documents and works by calculating N-dimensional lines or “hyperplanes” having the maximum distance with respect to the nearest data examples. The image below represents a visualisation of this process.



[150]

Figure 27: SVM visual representation

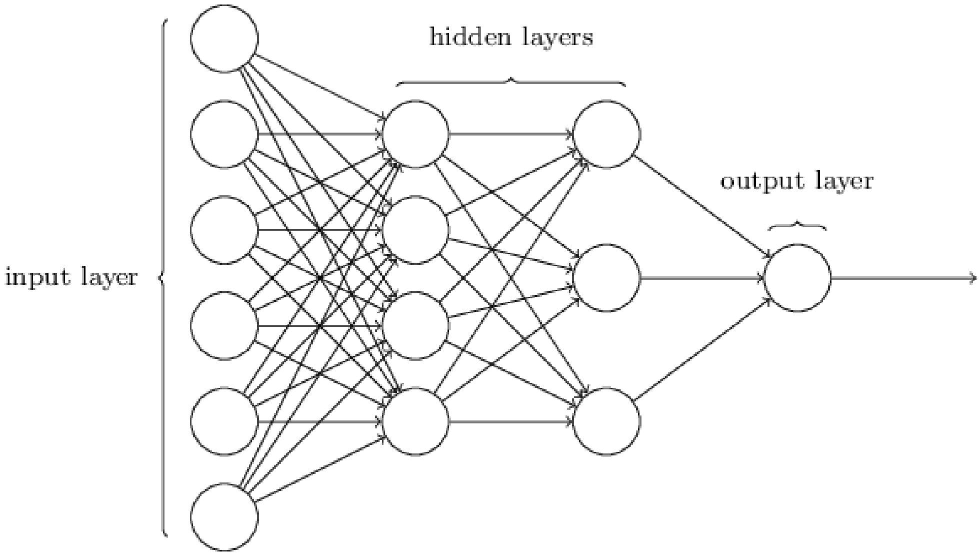
Random Forest

The third example of a machine learning technique for text classification are random forests. A random forest is a combination of many decision trees combined into one algorithm. A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute. Every branch of the model then represents an outcome, and each leaf node signifies a class label. The path from the root to the leaf then represents the classification rules. A random forest combines many variants of such decision trees into one model, resulting in a prediction using the average of the output of all trees. This makes it more accurate and reduces overfitting. However, it makes the method quite computer intensive and time consuming.

Neural networks

[150] https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/

A final example of text classification algorithms are neural networks. Compared to alternatives such as Naive Bayes, neural networks don't try to understand the texts, they just try to classify them. Neural networks are often used for chatbots. A neural network resembles a human brain by deploying elements that mimic human synapses and neurons. These are represented by nodes and weights that are carefully fine-tuned until the best possible prediction is made. The image below shows a basic representation of a neural network consisting of multiple layers, nodes, and strings.



[151]

Figure 28: Neural network representation

Training a neural network involves using an optimisation algorithm to find a set of weights to best map the inputs to outputs. The algorithm usually needs large datasets to do this and is prone to overfitting. However, a neural network can solve highly complex problems in a variety of application areas, making it suitable for NLP text classification.

[151] https://databricks.com/fr/glossary/neural-network

Stay in touch

@InteroperableEU) / Twitter

Interoperable Europe - YouTube

Interoperable Europe | LinkedIn

DIGIT-INTEROPERABILITY@ec.europa.eu

https://joinup.ec.europa.eu/collection/interoperable-europe/interoperable-europe

Publications Office