Machine Learning Approaches for Sub-surface Geological Heterogeneous Sources

Molood Arman

université PARIS-SACLAY

Machine Learning Approaches for Sub-surface Geological Heterogeneous Sources

Approches d'apprentissage automatique pour des sources hétérogènes géologiques de sous-sol

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° d'580, sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: Informatique
Graduate School : Informatique et sciences du numérique, Référent : CentraleSupélec

Thèse préparée dans la unité de recherche **Laboratoire interdisciplinaire des sciences du numérique (Université Paris-Saclay, CNRS)**, sous la direction de **Nacéra Seghouani**, Professeure, la co-direction de **Francesca BUGIOTTI**, Enseignant-chercheure, la co-supervision de **Sylvain Wlodarczyk**, chef de projet chez Schlumberger.

Molood Arman

THESE DE DOCTORAT

NNT : 2023UPASG014

Composition du jury

Esteban Zimanyi Professeur, Université Libre de Bruxelles	Rapporteur & Examinateur
Amel Bouzeghoub Professeure, Telecom SudParis	Rapportrice & Examinatrice
Hélène Bonneau Dr., Université Paris-Saclay, LISN	Examinatrice
Anne Vilnat Professeure, Université Paris-Saclay, LISN	Examinatrice

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Approches d'apprentissage automatique pour des sources hétérogènes géologiques de sous-sol

Mots clés: Apprentissage automatique, Entité Nommée, Modèle de langage spécifique à un domaine, géologie de sous-sol

Résumé: Dans le domaine de l'exploration et de la production du pétrole et du gaz, il est essentiel de comprendre les structures géologiques de sous-sol, tels que les diagraphies de puits et les échantillons de roche, afin de fournir des outils de prédiction et d'aide à la décision. Exploiter des données provenant de différentes sources, structurées ou non structurées, telles que des bases de données relationnelles et des rapports numérisés portant sur la géologie du sous-sol, est primordial. Le principal défi pour les données structurées réside dans l'absence d'un schéma global permettant de croiser tous les attributs provenant de différentes sources. Les défis sont autres pour les données non structurées. La plupart des rapports géologiques de sous-sol sont des versions scannées de documents. L'objectif de notre travail de thèse est de fournir une représentation structurée des différentes sources de données, et de construire des modèles de language spécifique au domaine pour l'apprentissage des entités nommées relatives à la géologie du sous-sol.

Title: Machine Learning Approaches for Sub-surface Geological Heterogeneous Sources

Keywords: Machine Learning, NER, Heterogeneous documents, Domain-specific language models, Sub-surface geology

Abstract: In oil and gas exploration and production, understanding subsurface geological structures, such as well logs and rock samples, is essential to provide predictive and decision support tools. Gathering and using data from a variety of sources, both structured and unstructured, such as relational databases and digitized reports on the subsurface geology, are critical. The main challenge for the structured data is the lack of a global schema to cross-reference all attributes from different sources. The challenges are different for unstructured data. Most subsurface geological reports are scanned versions of documents. Our dissertation aims to provide a structured representation of the different data sources and to build domain-specific language models for learning named entities related to subsurface geology.

To women of my homeland, Iran, who fight bravely and resiliently for their freedom and equality
Woman, Life, Freedom
Femme, Vie, liberté
Zan, Zendegi, Azadi

Acknowledgements

I would like to express my deepest appreciation to the members of my dissertation committee who gave me this honor by agreeing to serve on the committee - Esteban Zimanyi, professor at Université Libre de Bruxelles, Amel Bouzeghoub, professor at Telecom SudParis whom I am most appreciative for their time and extreme patience and also for their encouragement, scientific and constructive feedback. I also extend my deepest gratitude to Hélène Bonneau-Maynard and Anne Vilnat, professors at Paris-Saclay, for their time and consideration, knowing they would probably have less than two weeks to read my dissertation.

I would like to acknowledge and give my sincere and heartfelt gratitude and appreciation to a wonderful supervisor Nacéra Seghouani who made this work possible. Undoubtedly, completing this dissertation would not have been possible without her continuous support and deep understanding of my life's sudden and challenging circumstances. She always provided me with the guidance and counsel I needed to succeed in my path with her academic instruction, meticulous scrutiny, scientific perspective, and encouraging empathy. I would also like to thank Sylvain Wlodarczyk and Francesca Bugiotti for providing perspective, feedback, and practical suggestions throughout this research.

I also would like to thank Patrick Marcel from the university of Tours, Paolo Papotti from Eurecom in Sophia Antipolis, Philippe Caillou from the university Paris-Saclay and again Anne Vilnat from Paris-Saclay for their dedication, scientific perspective, and constructive advice.

This dissertation would not have been possible without financial support, which was fully funded by the Ministry of Higher Education, Research and Innovation, which has entrusted its implementation to the ANRT (Association Nationale de la Recherche et de la Technologie).

This work would have been much more difficult to complete without the support and friendship of some colleagues at the Schlumberger Montpellier Technology Center (MPTC). I am indebted to them for their emotional support and occasional advice.

Finally, but not least, I want to thank my parents, family, and friends. I want to express my gratitude to my parents for their unconditional love, support, care, and instruction in the moral values which guide my daily decisions, for trusting me when I would lose hope, and for being with me when I fell sick. Also, special thanks to Soudeh Ghasemian, a friend who stood by my side when times got hard and has never gotten tired of listening to my dramas and crying with me in the bad times. Thank you for being so much joy in my life and being my sister through it all, and making me laugh when I didn't even want to smile, and to Cristian Hedes, a friend on whom I can always depend and who was by my side like a brother in this prolonged journey.

Table of Contents

1 - Introduction

1.1 . Context

1.1.1 . Schlumberger

Schlumberger[2] is recognized as the world's largest service company for the oil and gas industry and has been operating in this field since 1920. A service company is a business whose income is from providing services instead of selling physical products [1]. A closer look at the oil and gas industry reveals that "service companies provide the infrastructure, equipment, intellectual property, and services needed by the international oil and gas industry to explore for, extract, and transport crude oil and natural gas from the earth to the refinery, and eventually to the consumer" (Wikinvest, n.d. [1]). Within the industry, The company currently has more than 92,000 employees across more than 140 different nationalities working in about 85 countries.

The main business of Schlumberger can be divided into two segments: oilfield services and IT services. "The oilfield services segment is a wide range of services and products covering formation evaluation, directional drilling, well cementing and stimulation, well completion, and productivity to consulting software and information management" [2]. On the other hand, the IT services segment provides services that support key industry operational processes. Over 100 years, the company has accumulated a wealth of oil exploration and production knowledge. Additionally, Schlumberger has demonstrated a commitment to using technology and innovation in processes through its extensive network of research and innovation centers. The company also derives a competitive advantage from its global perspective, providing a diverse workforce and international experience to support local knowledge and deliver superior service anywhere. Schlumberger is constantly working on research and development to get advanced approaches for its operations and processes. It aims to provide innovative technologies and solutions for oil and gas exploration from the reservoir sub-surface to the surface. The company makes significant investments in R&D because it sees technology as a long-term strategy to support and expand its operations. Schlumberger invests more annually in R&D than all other oil service companies combined [2].

1.1.2 . Sub-surface Geology Domain Terminologies

Experts deal more with sub-surface data than outcrop data (the rock details on the surface of the ground) while they work in the petroleum domain. The sub-surface geology domain studies the physical properties and location of rocks found below the earth's surface. Sub-surface studies can provide critical inputs for oil

1. https://www.shorturl.at/jkY14
2. https://www.slb.com/

and gas development and carbon sequestration. Sub-surface studies are essential in domains such as the petroleum and construction industries. For this industry, mining sand and gravel deposits are an important source of revenue. The decisions made throughout the mining process can be influenced by knowing how much and what kind of sand and gravel are present in a mineral deposit. In addition, sub-surface studies are critical to ensure a consistent supply of clean water; it is crucial to understand and safeguard groundwater aquifers, which is why sub-surface geology study is crucial. Other specific applications sub-surface investigations can offer are seismic imaging of magma chambers, geothermal exploration, identifying active faults (paleoseismology), and metal mining. There are terminologies and vocabularies specific to this domain. The definition of the most important ones, which helps for a better understanding of the content of this research, are explained in the following:

Stratigraphy. The word stratigraphy comes from the Latin word Stratum (layers of rock) and the Greek word graphia (to study). Stratigraphy is a geoscience branch that mainly studies rock deposits (strata) and stratification (buffer layers). Stratified rock can result from successive lava flows or the formation of extrusive igneous rocks [2] [3].

Stratigraphy has three related subfields: lithostratigraphy (lithologic stratigraphy), biostratigraphy (biologic stratigraphy), and chronostratigraphy (stratigraphy by age) [3]. Table 1.1 shows the principal stratigraphic unit terms related to this categorization.

Table 1.1 – *Stratigraphic categorization and their related unit terms [2]*

Stratigraphic categorization	Stratigraphic Unit Terms	Equivalent Terms
Lithostratigraphy	Supergroup, Group Formation Member Bed(s), Flow(s)	
Biostratigraphy	Biozones: Range zones Interval Zones Lineage Zones Assemblage Zones Abundance Zones Other kinds of biozones	
Chronostratigraphy	Eonothem Erathem System Series Stage Substage (Chronozone)	Eon Era Period Epoch Age Subage (or Age) (Chron)

3. https://en.wikipedia.org/wiki/Stratigraphy

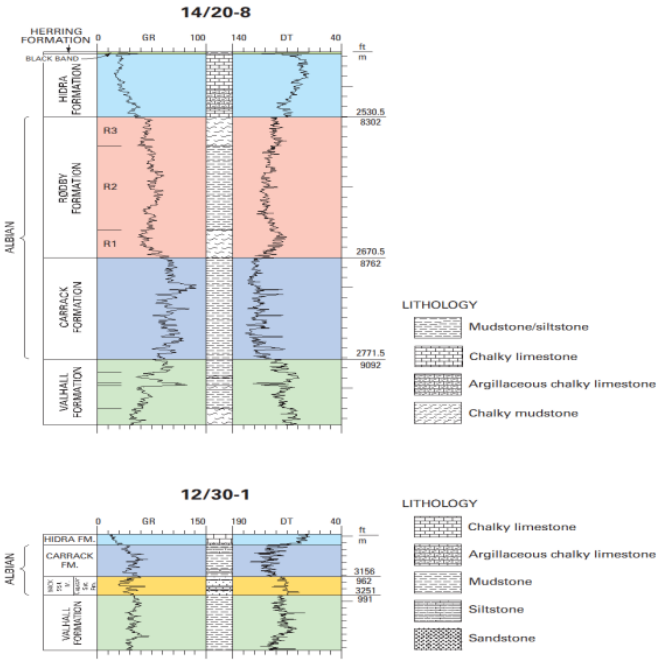


Figure 1.1 – *LITHOSTRATIGRAPHY: The Albian sequence of the North Sea [4]*

Lithostratigraphy. The stratigraphy part uses lithology (the type of rocks) and stratigraphical relationships to characterize and name rocks. This information is used to organize rock bodies into lithostratigraphic units. Rock type physical contrasts cause variation in rock units (lithology). Lithostratigraphy is the division of rock successions into units based on lithology. These Units can be identified by visible physical characteristics and are classified in the following order, highest to lowest [2] [3]:

- **Group** is made up of two or more adjacent or related formations. Related groups may be part of a supergroup.
- **Formation** is the primary unit of lithostratigraphy that is only identified on lithology.
- **Member** is a formation's designated lithological subdivision, which may extend into other formations.
- **Bed** is a member's or formation's named distinctive layer (a key bed or a marker bed).

The division, classification, and arrangement of rock strata according to their lithologic characteristics is known as lithostratigraphic classification.

It is necessary to know some critical points about lithostratigraphy in the subsurface domain:

- Lithostratigraphy is an essential component of model development in exploring oil and gas.

— The goal of the petroleum geologist is to create a comprehensive, all-encompassing stratigraphic model that aids in the exploration of oil and gas. In that model, lithostratigraphy is included.

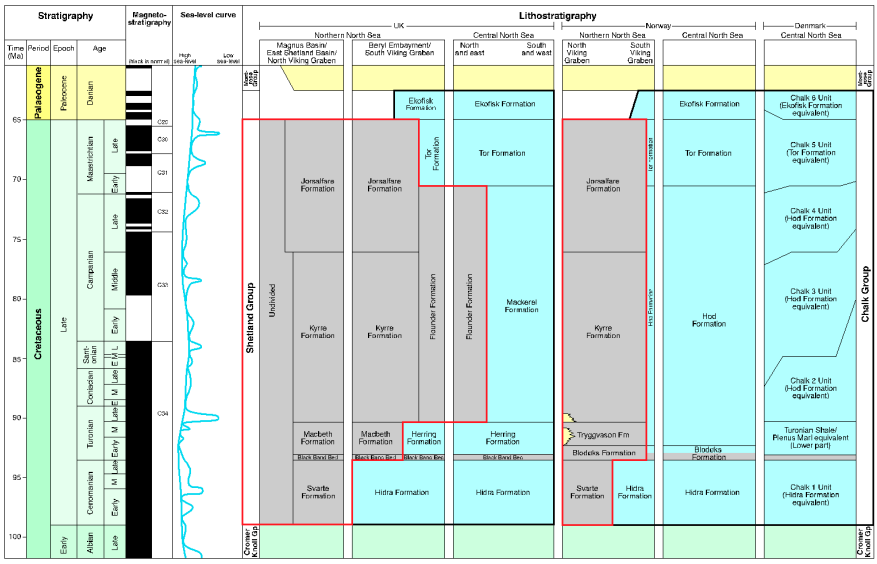


Figure 1.2 – *Upper Cretaceous lithostratigraphy correlation charts in different locations: UK, Norway, and Denmark [4]*

Lithology. The study of the fundamental physical properties of rock units, such as color, texture, grain size, and composition, is known as lithology. Lithology can refer to a thorough explanation of these traits or a synopsis of a rock's overall physical characteristics. Sandstone, slate, basalt, or limestone are also examples of lithologies.

Formation. The fundamental unit of lithostratigraphy is a formation, often known as a geological formation. A formation comprises a specific number of rock strata with similar lithology, facies, or other characteristics. Since the thickness of the rock layers that make up a formation is not a defining factor, the thickness of various formations can vary considerably [3].

Boundaries of Lithostratigraphic Unit (Depth Interval). Boundaries of lithostratigraphic units are fixed arbitrarily within zones of lithologic gradation or at naturally occurring acute or distinct contact between different lithologies at places of lithologic change. These boundaries of the lithostratigraphic units define the depth interval of that lithostratigraphic unit in between two units [2] [3] (as shown in Figure 1.1).

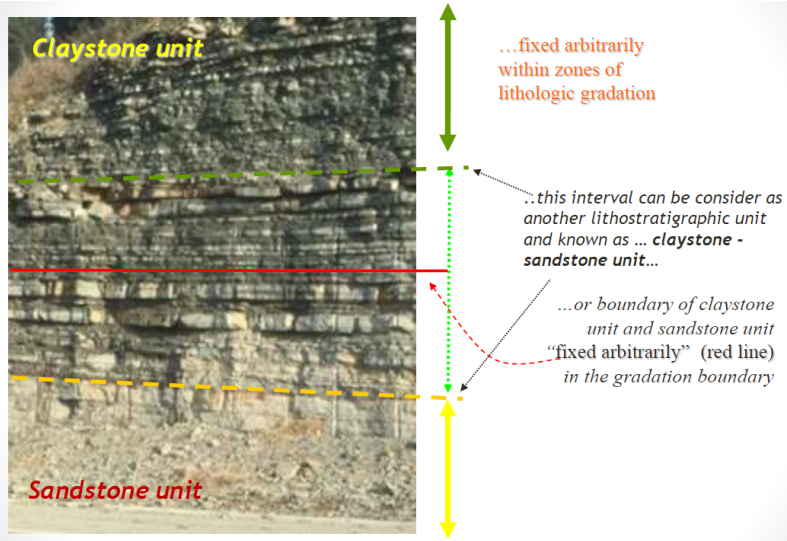


Figure 1.3 – *Boundaries of Lithostratigraphic Unit (Depth Interval) [4]*

Well Definition. A well is a hole planned or already bored into the earth to enable the interchange of fluids between a subterranean reservoir and the surface (or another reservoir) or to enable the detection and measurement of rock properties. A physical well is formed each time the drill bit pierces the earth's surface. There are particular well-related elements that aid in better defining a well, including the following [5]:

1. **Well Origin:** The location on the ground's or ocean's surface is known as the well origin, where the drill bit enters or is intended to enter the ground to construct or overhaul a well.
2. **Wellbore:** The wellbore is the route that drilling footage takes from the well origin (at the top) to its end (bottom).
3. **Wellbore Segment:** A well's wellbore segment is a distinct interval that extends from the well's starting point (well origin) to its ending point or extra distance from a point in an existing wellbore to the end (terminating point).
4. **Wellbore Contact Interval:** To produce, inject, or provide service, a wellbore must come into contact with one or more stratigraphic zones within a defined depth range known as the "wellbore contact interval".
5. **Wellbore Completion:** A group of one or more wellbore contact intervals that combine to create or inject fluids is known as a wellbore completion.
6. **Wellhead Stream:** A fluid flow through a conduit called the "Wellhead Stream" is controlled by an installed wellhead arrangement.
7. **Kickoff Point (KOP):** It is a location where directional operations start in a vertical wellbore or inclined section of a slant well.

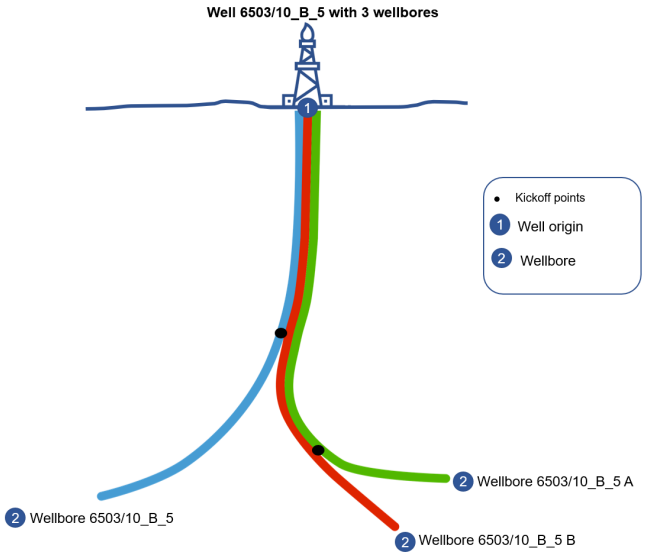


Figure 1.4 – *A well with three wellbores*

1.2 . Problem Statement

The drilling process for discovering new productive oil and gas wells is becoming a highly investigated task in oil and gas companies. Drilling is an expensive, time-consuming, and very environmentally destructive activity. Onshore, a single well costs 5-8 million dollars, while deep-water wells cost 100-200 million dollars or more [6]. Before and during drilling, oil and gas companies gather and investigate data about the earth's sub-surface before and during drilling to drill a profitable well properly. Data is gathered from regional to microscopic levels. This information helps oil and gas companies find more productive wells and avoid drilling unproductive ones. Many geological reports have been accumulated during exploration, development, production, and other wells related procedures, each containing geological themes: geological surveys, cutting reports, end-of-well reports, and core lab reports. Each of these different kinds of reports talks about separate analyses such as the geological analysis, the study of material extended from the mud during the drilling process, the formation and similar information when the well is drilled, or the core extraction analysis of the formations in the laboratory. The contents of these reports are stored in various formats, including PDFs, images, tables, and structured databases. These reports contain many structured and unstructured data [7]. Structured data are usually stored and managed using relational databases; however, gathering and referencing the different values to the same entities from heterogeneous structured data sources is difficult due to a lack of a global schema in this domain. Unstructured data include diverse types of information, which can have more potential value than structured data. Therefore, many research types are focused on developing methods of efficiently managing, mining,

and using these unstructured data. Geoscience literature is a rich resource that can facilitate knowledge discovery and information extraction. This literature contains a large amount of meaningful information that defines data expertly and can be applied to train new models and enrich our understanding [7] [8]. To understand these sub-surface geological reports, we need first to find a way for a computer to understand and store the structure of these reports in a way that humans see them. Secondly, we need to use NLP methods to help machines to understand the language/texts like humans understand them. To extract the information from the text of these sources, we need to apply NLP tasks such as named entity Recognition (NER). Self-attention-based language models are the most influential models in NLP domains for some downstream tasks, such as machine translation or summarising text. However, there are still lots of shortages when it comes to domain-specific NLP downstream tasks.

1.3 . Proposed Approaches: Specific Aims

The specific aims of this dissertation are related to the three challenges mentioned above. Given the available structured and unstructured data, we want to investigate approaches to understand the sub-surface geological domain better. Then we could associate each type of data source (structured/unstructured) with different challenges as listed above:

1. **Part I:** Deal with heterogeneous structured data sources.
    - — Gathering and merging heterogeneous structured data sources: automatical-discovery of the same attributes with different names.
2. **Part II:** Deal with heterogeneous unstructured data sources.
    - — Recognize the structure of heterogeneous unstructured sources (such as PDFs and scanned documents).
    - — Perform domain-specific entity recognition to accelerate the information retrieval process from heterogeneous unstructured sources.

1.3.1 . Contributions

For each specific challenge, we conducted various studies. The available dataset comprised about 44,000 heterogeneous structured data sources and about 3,400 heterogeneous unstructured data sources. These studies resulted in the following contributions; organized according to the chapter, they are illustrated in detail.

1. **Part I:** Dealing with heterogeneous structured data
    - PROCLAIM: An Unsupervised Method to Build a Domain-specific Global Schema
        - — Introducing the concept of attribute profile by taking into account the data type using: (i) the statistical distribution and the

dimension of the attribute's values and (ii) the name and textual descriptions of the attribute. These properties give a unified representation to each attribute.

- — Defining Extended-OPTICS, an extended version of OPTICS that clusters the attributes profiles by introducing a dynamic minimum number of points in place of a static one.
- — Proposing the concept of labeling function by taking into account the following: (i) the attribute descriptions, and (ii) the attribute names for each cluster to automatically assign a label to each one as the final name of the similar attributes in the global schema of the domain.
- — Presenting PROCLAIM, a domain-independent method for schema label prediction. We evaluate our method on two different datasets from two different domains, and PROCLAIM performs very well with increasing heterogeneity of the datasets.

2. **Part II:** Dealing with heterogeneous unstructured sources
   - Machine Learning for Document Structure Recognition
     - — Introducing OCRANA, a scalable framework that efficiently transforms heterogeneous PDFs or image documents, processed by different OCR engines, into unified structured information to prepare them for further analysis.
     - — Demonstrating the unified data model of OCRANA, which allows representing different kinds of structures of texts and their visual and content-based properties in an element-level scheme.
     - — Presenting position-based Naïve Bayes algorithm to find the defined layout labels for each line of heterogeneous PDFs or image documents presented in OCRANA. This model was built on a large-scale dataset generated by a weakly supervised approach.
     - — Showing empirical evidence that OCRANA can accelerate information retrieval from unstructured documents.
   - Unsupervised NER By Automatic Generation of Gazetteer
     - — Proposing GAGNER, a novel unsupervised approach to generate domain-specific gazetteers.
     - — Demonstrating that GAGNER only uses the corpus text as its input to generate gazetteers and showing that external resources (such as Wikidata knowledge base) are not needed. This approach is promising for any low-resource domain.
     - — Presenting that GAGNER can tag the less-known abbreviations and the wrong written forms (typos) of the words in noisy corpora,

making it helpful to annotate a corpus, especially for a generated corpus from the output of OCR engines which usually do not have very high-quality texts.

- Highlighting that GAGNER uses vector word representations to find similar words in terms of implicit semantic and/or syntactic information to generate a different group of entities in the form of gazetteers.
- Creating a training dataset using generated gazetteers to annotate the corpus to build a NER system by neural models. In other words, the final NER neural model builds a NER system using minimum resources.

- GeoBERT: NER using Domain-Specific Language Model
  - Presenting GeoBERT, a domain-specific BERT-variant language model for the sub-surface geology domain.
  - Adapting generic BERT model to a specific domain containing a limited number of sources.
  - Demonstrating that the generic BERT model can benefit from integrating an extension module to solve a specific domain's unseen vocabulary (OOV) issue in a limited-source sub-surface domain.

1.4 . Thesis Outline

This dissertation comprises four main chapters, including an introduction and a conclusion chapter. Each main chapter presents a complete approach for each challenge presented in Section 1.3. We summarize each chapter in the following. The dissertation then concludes with Chapter 6, which discusses the conclusions and future work.

1.4.1 . Chapter Two

In this chapter, we present PROCLAIM (PROfile-based Cluster-Labeling for AttrIbute Matching), an unsupervised method for matching attributes from a large number of heterogeneous sources in a specific domain. Our results show that PROCLAIM is a practical, fully automatic method to discover a set of meaningful vocabularies which are the backbone of the definition of a specific domain. PROCLAIM defines the concept of attribute profile by taking into account the data type using: (i) the statistical distribution and the dimension of the attribute's values and (ii) the name and textual descriptions related to the attribute. The cluster-labeling function inputs these properties to automatically assign labels to many attributes [9].

PROCLAIM gives a name (label) to each group of similar attributes from different schemas, which can represent the essence of each group. The set of these labels defines the global schema.

1.4.2 . Chapter Three

This chapter presents OCRANA (Optical Character Recognition ANAlytics), a scalable framework that efficiently transforms heterogeneous PDF or image documents processed by different OCR engines into unified structured information. OCRANA relies on a data model that represents both intermediate and final results. OCRANA is based on a data model which supports heterogeneous sources. The data model is designed to keep different kinds of structures of texts and their visual and content-based properties (tables, headers, ...). Moreover, the scalable architecture of the framework enables storing the results in a columnar data format and processing a large number of documents, allowing the application of analytics and information labeling effectively. We also present a position-based Naïve Bayes classifier algorithm that efficiently recognizes the structure of documents based on the OCRANA data model.

1.4.3 . Chapter Four

This chapter proposes a named entity recognition (NER) system based on the automatic generation of name gazetteers. GAGNER (GAzetteer Generation for Named Entity Recognition) automatically generates the lists of entities (gazetteers) to create an annotated named entity corpus. GAGNER can tag very noisy corpora containing many typos made originally by humans in the texts and/or made by OCR engines due to low-quality outputs while extracting the texts from scanned images of documents. The system can handle different named entities (NE) types targeted by domain experts. The GAGNER approach uses shallow neural network methods to generate the corpus-related gazetteers automatically. We apply a pre-trained language model (BERT) on the annotated NE corpus to train a customized name entity recognition model based on our corpus. Since the evaluation results of generated gazetteers show a high accuracy, we can be sure that the annotated corpus has very high accuracy. We experimentally evaluate the generated gazetteers on the sub-surface geology domain corpus as the output of dense part of reports (around 34,000 scanned images, PDFs, and documents) labeled in Chapter 3 containing around 634,000 sentences. We manually evaluate our NER model using the BERT model on the high-quality annotated NE corpus. The evaluation results of having a NER system in this specific domain are promising.

1.4.4 . Chapter Five

This chapter presents GeoBERT, a domain-specific BERT-variant language model for the oil and gas industry. BERT is a pre-trained language model which offers subword representations rather than word-level representations for representing both the tokens of input text and the output. The principal shortage of a generic BERT model in a new domain is related to the OOV (out-of-vocabulary) words when many of the common words (most frequent words) in a new domain are OOVs, and there is a lack of exact embedding representation for them. GeoBERT adopts the BERT model's embedded knowledge by injecting the domain-specific

vocabularies, OOVs, into the BERT model. Our findings indicate that this approach is consistent with using small domain-specific sources. According to our knowledge, this approach has never been used before in the geology, oil and gas domain, or sub-surface domain, let alone on limited resources. The final result of the NER task in the sub-surface geological domain shows a significant increase in precision and recall compared with the generic BERT model.

2 - PROCLAIM: an Unsupervised Method to Build a Domain-specific Global Schema

2.1 . Introduction

A time-consuming step in integrating the data sources is finding matching attributes across heterogeneous data sources [10]. In this context, schema matching is the process that automatically creates a global view of various independently developed schemas [11]. The integration process to generate a global schema must identify relationships between equivalent schema elements of the data sources. Data sources will inherently have diverse schema representations modeled separately by different people. Schema integration must reconcile this diversity in schema representation to establish schema entity mappings. Once schema mappings are developed, mapped elements of the data sources are combined to form a unified global view. For example, schema integration must integrate the search results fetched from diverse data sources to offer a unified search result. The main problems that we encounter with data at Schlumberger are as follows:

— **Heterogeneous sources:** Data heterogeneity leads to input from diverse sources into a unified system. In every data integration strategy, a primary concern is a different available schema for various sources, leading to the redundancy of the same attributes with different names. Also, it can cause data volume growth since data duplication can accrue a lot. As the starting point of our research, one problem was the integration of more than 40000 heterogeneous data sources. The relation between different schema elements of sources was unclear, which led us to this central question of how we want to gather and store data. The metadata and description for these data sources' schema are unavailable. Also, we did not have any reference (mediated schema) for the schema integration, another problem that will be explained in more detail later.

— **Noisy real datasets:** Not all unstructured data has high quality, incredibly raw data that can be quite uneven in quality. The lack of consistency in quality happens because data is difficult to verify and, consequently, is not always accurate. For example, different features related to the same entities can be captured and stored in different databases. The units of some measured values can be the same, but they can be stored in different measurement systems or formats (e.g., time and dates). In this case, finding duplicate records is not an easy task. Also, much of the data may not be reliable because of a human data entry error or missing values. Regarding the massive increase in big data, tackling these challenges is time-consuming and expensive. Most machine learning methods need clean datasets to have

good results in accuracy and precision for further analysis.

— **Lack of a global schema:** schema matching is not a new research domain. Despite over two decades of extensive research, schema matching still seems to involve ad-hoc solutions that do not follow any standard [12]. . According to the classical definition, a pairwise match is applied between the schema of two different datasets. In this case, if we have a global schema as the mediated schema, all other schemas are matched by the mediated schema, and correspondence attributes can be found. However, in the oil and gas domain, since the data is gathered from different heterogeneous sources related to various companies and even in different countries and continents, a global schema does not exist. To our knowledge, most existing tools and models for schema matching needs a mediated schema. Then we need a method that can match attributes to find the global schema for our domain without mediated schema, which leads us to choose clustering approaches.

We present PROCLAIM (PROfile-based Cluster-Labeling for AttrIbute Matching), an unsupervised method for matching attributes from many heterogeneous sources in a specific domain. This chapter presents empirical evidence using different datasets showing that our method is efficient on significant heterogeneous sources. As a final output, PROCLAIM can automatically create a set of unique labels assigned to a high percentage of attributes from all attributes coming from different heterogeneous sources. The main contributions of our research are:

1. Defining the concept of attribute profile by taking into account the data type using: (i) the statistical distribution and the dimension of the attribute's values and (ii) the name and textual descriptions of the attribute. These properties give a unified representation to each attribute.
2. Defining Extended OPTICS, an extended version of OPTICS, introduces a dynamic minimum number of points instead of a static one to cluster the attribute profiles.
3. Defining the concept of labeling function by taking into account: (i) the attribute descriptions, and (ii) the attribute names for each cluster to automatically assign a label to each one as the final name of the similar attributes in the global schema of the domain.
4. Proposing a domain-independent method for schema label prediction. We evaluate our method on two different datasets from two different domains, and PROCLAIM performs very well with an increase in the heterogeneity of the datasets.

We use domain experts to validate the automatically chosen labels for each attribute. Our evaluation indicates that the quality of these automatically generated labels is very promising.

This chapter is organized as follows: Section 2.2 reviews the related studies on schema matching. Section 2.3 presents a brief overview of PROCLAIM. Sections 2.4, 2.5, 2.6, and 2.7 detail each building block of PROCLAIM. Section 2.8 illustrates the results of our experiments in two different domains. Finally, Section 2.9 presents the contributions and conclusion.

2.2 . Related Work

A schema is a group of elements connected by some sort of structure, including element types, attributes, basic types, etc. [13] [14] [15] [16]. Schema matching is a crucial phase in the data integration process. Finding correlations between schema is the main goal of the schema matching process, which will be helpful later during the data integration phase. For a long time, the schema mappings had been a manual task. However, with the advent of big data, schemas became more extensive and complicated, leading to the time-consuming and error-prone manual design of mappings [17]. Therefore, presenting automatic schema matching became one of the attractive domains of research. Schema matching's primary focus is determining the attribute correspondence across syntactic and semantic heterogeneity in data sources to support the merging decision. The availability of multiple and heterogeneous data sources has recently given new perspectives to the schema-matching problem. Schema matching is not a new research domain. Still, after decades of research [18] [19] [20] [21] [22], there still needs to be a standard way, especially when we think about specific contexts or domains where data schema and representations are heterogeneous. There are various types of schemas, including conceptual domain and relational database schemas. For instance, standards like DTD, XML Schema, and Relax NG may specify XML schema. But neither the diversity of schemas nor their representations ought to stand in the way of a general discussion. Much research has been done in the literature on using data source instances to recognize the correspondence between attributes during schema matching [15]. In the literature, defining the global (mediated) schema, also known as a target schema, is the first step in the schema-matching process [13].

Figure 2.1 shows a taxonomy of schema-matching approaches. An implementation of matching may use more than one matching algorithm [23]. There are three categories of information typically used to solve the problem of schema matching, e,g. identifying the semantics of database attributes and identifying the correspondences between database schemas, e.g., 1) schema information, 2) instances, and 3) auxiliary information [15] [23] [24]. A number of solutions have been suggested for schema matching. Based on the available database information. Various approaches have been proposed for different levels of this information, some of which rely on using each level independently as identified individual matching algorithms. At the same time, other approaches involve a combination of individual matching

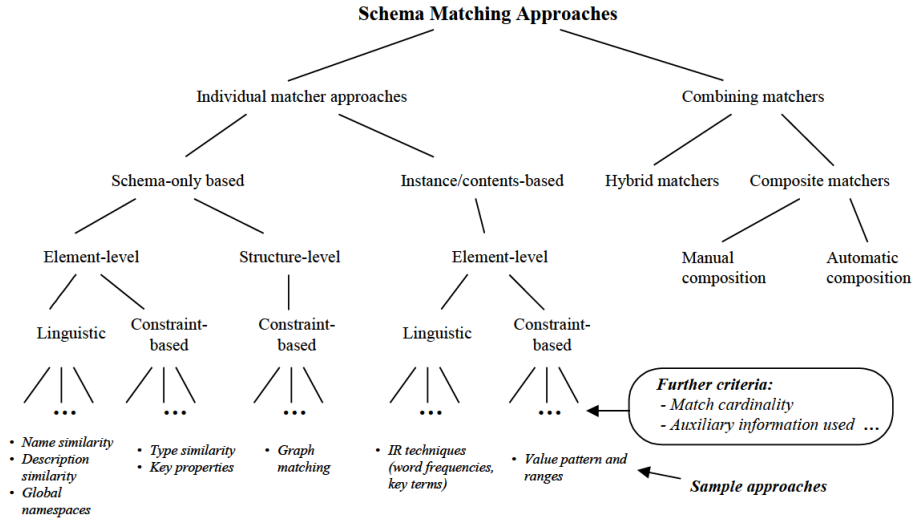


Figure 2.1 – *Classification of schema matching approaches [23]*

algorithms to boost the matching result size. Three levels of schema information have been identified: the schema level, the instance level, the hybrid level, and the auxiliary level [15]. Identifying the association between the attributes utilizing dataset values and semantic and syntactic rules to determine the correspondence between attributes throughout the schema-matching process is the main goal of many published works [15]. The majority of works on schema integration presupposed a global (mediated) schema. They then tried to discover a solution for better matching, mainly a pairwise matching between the source and mediated schema. In this situation, it is difficult to construct a global schema that corresponds to all the attributes of a specific domain [25]. Below is more information on these levels of schema:

— **Schema Levels:** Three levels of information are included at the schema level: the linguistic level, the constraints level, and the structural level. The **Linguistic level** makes use of meta-data details such as the names or abbreviations of the attribute and any textual descriptions that are readily available. The data types of database properties like string and numeric, instance ranges, and distinct types all have a role in the **Constraint level**. The schema's internal and external structure and cardinalities between keys, like primary and uniqueness, are used at the **Structure level**.

— **Instance Level (known as Content Level):** It is frequently challenging to obtain information from the Schema structure because it is either unavailable or cannot be used for matching. Instances are thought to be the most effective and trustworthy source of information for determining the similarities and associated attributes of a schema by using the characteristics of the available values and instances.

— **Hybrid Level:** The hybrid level collects information from both the instance level (values/instances) and the combination of the schema metadata, such as attribute names, data type, structure, and description.

— **Auxiliary Level:** Auxiliary level combines information from the existing schema with new information collected from external sources. WordNet and dictionaries are examples of external sources that can help determine the semantic connections between names or abbreviations of schema attribute names, such as synonymy and hyponyms if they are comparable.

Furthermore, we need to talk about the leading alternatives for matching the granularity and cardinality of schemas. There are two main options for the granularity of schema matching: element-level and structure-level matching [15][23] [24]. When using *Element-level matching*, each element from the second input schema's matching elements is detected for each element of the first. Only items with the highest level of granularity, such as attributes in XML or columns in relational databases, are taken into account. On the other hand, matching at the *Structure-level matching* refers to matching sets of elements that coexist in a structure. In the best-case scenario, every element of the two structures would be matched. Alternatively, only some components may be required to match (e.g., a partial structural match). The need for partial matches sometimes arises as subschemas of different domains are being compared. For example, "oil-based mud" is a drilling fluid used in drilling engineering, which is expensive but is worth the cost. "oil-base mud" value may come from non-water-based drilling fluid, while "water-mud" comes from water-based drilling fluid (WBFs) data sources. Real-world data is also frequently noisy, requiring data cleaning for most integration methodologies. However, data cleaning is expensive and time-consuming when it comes to massive data. This chapter develops a heuristic method (PROCLAIM) to deal with the real world and massive data.

In recent years, generating schema labels through dataset content analysis is becoming trendy [22] [26] [27]. With no standard for schema attributes names, ambiguous names are often found in real-world datasets (e.g., many attribute names in different schema contain abbreviations and compound nouns that hinder automated schema matching). In [22], they propose a supervised method that recommends alternative schema labels for ambiguous names of attributes by considering the content of descriptions and metadata. In [26], they proposed a model based on the word embeddings of the table attributes tokens using every table's contextual information to predict the similarity score of each pair of query-table to retrieve the related table. In [27], they proposed a context-aware schema-matching model for predicting labels of attributes without headers by using word embeddings and language models. PROCLAIM automatically generates schema labels by considering the attributes' context and values.

Table 2.1 shows how PROCLAIM implementations fit the classification criteria introduced above.

Table 2.1 – *Characteristics of proposed schema match approach (PROCLAIM)*

Type	Metadata	Granularity	Schema-level	Instance-level	Auxiliary info.	Approach
Relational	Hybrid External Resource	Element-level	Linguistic Constraint	Linguistic Constraint	Dictionary	Hybrid

2.3 . PROCLAIM Overview

Schema matching aims to discover semantic correspondences of schemas attributes across heterogeneous sources. We aim to get a global attribute schema for all the independently developed schemas of the same domain. This process can be formalized as follows:
Given a set of schemas $\mathcal{S}$=$\{S_1, S_2, ..., S_n\}$ and the set of all attributes $\mathcal{A}$=$\{A_1, A_2, ..., A_n\}$ belonging to these schemas, each A_i contains the whole set of attributes $(a_1, ..., a_m)$ ($a_i \in A$ where $i \in [1 : m]$) used in the schema S_i. *Schema matching* selects sets of n-ary *mapping attributes* from different schemas that define a cluster of similar attributes $\mathcal{C}$=$\{C_1, C_2, ..., C_j\}$, as illustrated in Example 1.

Example 1. *Consider three schemas as a set of attributes about rental cars descriptions:*
S_1= {*Fuel_Type, Location, Mileage, Name, Price, Year, Transmissio*}
S_2= {*Country, Disp., HP, Mileage, Price, Type*}
S_3= {*fuel_type, maker, manufacture_year, mileage, model, price_eur, transmission*}
Also, consider the following attribute matches among the schemas:
C_1= {*Fuel_Type, fuel_type, fuel, fuelType*}
C_2 = {*Location, Country, city, county_name, state_name*}
C_3 ={*Name, maker, brand*}

All attributes are trivially a cluster by themselves. A *label l* can identify in the best way the essence of a semantic cluster of attributes. A *labeling function* $\mathbf{f}$*(C)* indicates the required process to define the label ($\mathbf{f} : C \rightarrow L$), where *L* is a set of labels ($l_i \in L$) and *C* is a set of similar attributes' group ($C_i \in C$). The set of labels L identifies the elements of a global schema for the given set of schemas $\mathcal{S}$, as illustrated in Example 2. This resulting schema is also the mediated or target schema.

Example 2. *Consider three clusters of similar attributes from Example 1. Based on the defined labeling function* $\mathbf{f}$*(C), labels* {*Location, Brand, Fuel, ...*} *will be assigned to each cluster of attributes automatically:*

C_1= {*Fuel_Type, fuel_type, fuel, fuelType*} $\rightarrow l_1$=$Fuel$
C_2 = {*Location, Country, city, county_name, state_name*} $\rightarrow l_2$=$Location$
C_3 ={*Name, maker, brand*} $\rightarrow l_3$ =$Brand$

The set of these assigned labels to each cluster of similar attributes defines the global schema for a specific domain: L= {*Fuel, Location, Brand, ...*}.

The main question addressed in this research is how to define an automatic process that discovers a set of labels that can effectively represent a global attribute schema for a specific domain [9]. The PROCLAIM method is proposed as an answer to this question. PROCLAIM is a new approach that enables automatic holistic schema matching, which leads to constructing a global attribute schema for a specific domain. Let us illustrate the procedure by following the main steps it involves, with the help of Figure 2.2:

1. a set of heterogeneous sources with different schema ($\mathcal{S}$) is provided as input;
2. the data from all sources are stored in columnar format storage;
3. the data type of each attribute is identified, and data with the same data type are stored in the same set ($\mathcal{A}_d$);
4. an attribute profile is computed based on the specificity of each data type ($\mathcal{A}_d$). This attribute profile can contain at most four properties (statistics, description, unit, and name property). Then each profile of attributes can contain at most four properties. The assigned profile to each attribute will be converted into a numerical vector;
5. an automatic labeling process is defined to find all similar attributes and gives a unified name to each of them. This process includes two principal components: (1) finding the most similar attributes from different schemas and (2) giving an automatic label to each attribute by a defined labeling function ($\mathbf{f}$). A density-based clustering algorithm will be applied to the numerical profiles to find the most similar attributes. Each profile vector represents a unique attribute;
6. the list of automatically computed labels will define a global attribute schema for a specific domain.

As explained in detail in the following sections, PROCLAIM can be applied to noisy real-life data. The method is designed to handle many heterogeneous schemas and proposes unified numerical profiling of information of any data type. The approach enables the usage of common machine learning algorithms such as clustering. Finally, the automatic labeling and merging of clusters allow the definition of a global schema that represents the synthesis of the heterogeneous schemas.

2.4 . Data Preprocessing

Some of the building blocks of PROCLAIM are initial steps to prepare the original datasets. Three main steps are defined as the initial steps in data preprocessing of PROCLAIM to ensure high-quality data (1) data integration, (2) data cleaning, and (3) data transformation.

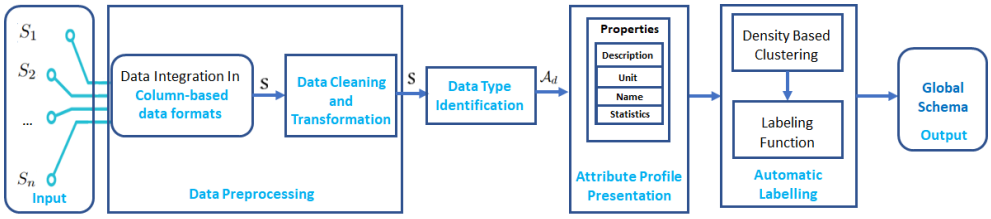


Figure 2.2 – *The framework of PROCLAIM to discover a global schema*

2.4.1 . Data Integration in Column-based Data Formats

Traditional databases store data sequentially row-wise (each row presents an individual record). In a columnar database, the values of attributes related to a record are present as a pair of attribute-value in each row. Since column-based data formats enable extensive parallelization and efficient partitioning techniques, they process large datasets effectively. Since column values are compressed into each data page, the I/O cost is reduced because the data in a single column is homogeneous. Since compressed columns are frequently so little that they can be totally cached in memory, queries are executed remarkably fast. Data access operations are usually over individual rows and show the best performances when retrieving only a subset of the attributes of a table when datasets are sparse and contain lots of empty values. Moreover, column-based data formats process big datasets efficiently since they provide large-scale parallelization and effective partitioning strategies. PROCLAIM, for its calculation, needs a tuple for each attribute's value showing its name and value. In this case, storing the data in a columnar-based format is much more efficient. Some of the advantages of this storage vs. row-based format storage are described in the following [28]:

1. Column-based storage is most useful when queries that need only a subset of attributes over big data are performed; Row-based storage is most useful when many of the attributes related to a record are wanted to use, or many records are needed to access.
2. Column-based storage is most useful for sparse datasets where many empty values may appear; The empty values can be easily separated from attributes with values, hence handling the missing value is so easy.
3. Big data sets frequently consist of hundreds to thousands of files and can even contain millions of records. Furthermore, these files are usually generated continuously, and efficiently processing such datasets requires large-scale parallelization for performance. Column-based storage will be progressively prone to splitting into independent jobs if the query calculation is associated with a single column at once. Big data can be split in a customized way, and we can recall one or more split chunks for further analytical queries.
4. In simple words, data compression can be defined as reducing the size of a file. The quantity of information required for data storage or transmission

is reduced through data compression. It simply saves money and time by lowering the number of resources required for data transmission and storage. Comparing the two formats, the columnar-based one can achieve higher compression rates. More effective compression is possible by storing values by column and placing those of the same type close to one another.

5. Data in real life may come in any format, such as CSV, JSON, and XML. Converting data from different formats to the one ideally appropriate for a specific need can be a process with a strenuous effort which might comprise detection, evolution, or modification of schemas, combining or splitting data sources, and applying to partition. Converting the incoming formats to columnar-based format can make this process much more manageable.

2.4.2 . Data Cleaning and Transformation

Every time data is obtained from various sources, it is gathered in a raw format that makes analysis impossible. Some actions are required to ensure data quality, such as handling missing values, redundancy, correction, and format transformation. Consolidating data involves removing pointless fields, columns, and records. Missing values by integrating data in columnar format have already been managed. Data might also be transformed: Data related to time or date should have the same format, or the well IDs (well identifiers) of different sources should follow the same format.

2.5 . Data Type Identification

When the search space is large (the number of attributes or schemas is significant), matching the complete input of schemas may require long execution times, and achieving high-quality results may take much work. One way to reduce the search space is to find similar attributes within the same data types. The heterogeneous sources provide attributes in different data types. Since the type of the attributes may not be provided in the metadata of sources, we need to identify the types given the values. One main problem in this step is that the original datasets need to be cleaned. We have to consider the type based on the data type of most instances (values). Here, we consider five data types, but this set can be extended if it is necessary:

— NUMERICAL representing all attributes whose value contains integer or float.

— CATEGORICAL containing all strings, characters, and mixed data type.

— DATE representing date and time such as DateTime and timestamps.

— RARE classifying attributes which have less than ten instances.

— UNIQUE refers to attributes with unique values (cardinality equals 1).

Formally, let d be the data type of an attribute a_i with probability $p \geq threshold$ (e.g., $threshold = 0.8$) where $a_i \in \mathcal{A}$ and $d \in \mathbf{D} = \{$ NUMERICAL, CATEGORICAL, DATE, RARE, UNIQUE$\}$ where d $\in$ [1,5]. *Data Type function* ($\mathbf{f}_D$) ($\mathbf{f}_D : \mathcal{A} \rightarrow \mathbf{D}$; $\mathbf{f}_D(a_i) = d$) pre-classified the attributes of the whole dataset into a maximum of five categories ($\mathcal{S}_d$), which contain attributes with the same data type.

$$\forall a_i, a_j \in \mathcal{A}_d : \mathbf{f}_D(a_i) = \mathbf{f}_D(a_j);\ \text{where } \mathcal{A}_d \subseteq \mathcal{A} \tag{2.1}$$

2.6 . Attribute Profile Representation

Once we have all the attributes belonging to the same data type ($\mathcal{A}_d$), we can group them to discover attributes coming from different schemas which contain the same information (e.g.,{name, maker, brand} in our example). PROCLAIM performs clustering and labeling based on the computation of a similarity matrix of numerical profiles of attributes. Before applying our algorithm, we must convert an attribute to a numerical profile based on its data type. A maximum of four components characterize any attribute in our representation according to the data type to which it belongs. These components are description, unit, name, and statistics. In this section, we describe each profile component and its contribution to analyzing the attributes classified in any of the six data types introduced in the previous section. Notice that the RARE type attributes are ignored due to the impossibility of computing a valid statistic.

Description Property The majority of datasets have a descriptive part for the schema where the meaning of each attribute can be found. In other cases, the description is not provided, but the used values belong to domain-specific terms or abbreviations, and this description can be retrieved, for example, using domain-specific Wikis. To create the description profile, first, we remove the stop-words, then apply the stemming method over a bag of tokens. Then for each description, the stems and the occurrence of each term (in all the different descriptions for any specified attribute) are used to build the description profiles. Removing stop-words in a specific domain is necessary since these words can appear in almost all descriptions and cause false similarities (e.g., for the domain of cars, the words such as car, vehicle, and automobile are the domain stop-words). We then transform the descriptions into categorical variables. Next, feature engineering is required to encode the different categories into a suitable numerical feature vector. One-hot encoding is a simple but efficient widely-used encoding method [29]. An example of converting categorical variables for some attributes to numerical values can be seen in Table 2.2.

Table 2.2 – *One-hot encoding for converting descriptions to numerical feature*

Attribute	displac	volum	engin	cc	repres	kw	ccm
ENGINE_DISPLACEMENT	0	0	0	0	0	0	1
ENGINE_POWER	0	0	0	0	0	1	0
DISP.	1	0	1	0	1	0	0
ENGINE	1	1	1	1	0	0	0

Unit Property Dimensions and units are fundamental tools to explain the characterization of phenomena [30]. A dimension measures a physical variable by fundamental quantities without numerical value, such as distance, time, mass, and temperature. However, a unit is a specific way to assign a measurement (with numerical value) to the dimension, e.g., a dimension is a length, whereas meters or feet are relative units that describe length [30]. Dimensions and units are commonly confused, even though the solution to most problems must include units. The distribution of the same entity in different units can be shifted, but by consideration of the same dimension, the similarity of shifted distribution can be found. Attributes with units related to the same dimension are also related to each other through a conversion factor. For example, Kelvin or Celsius measures the temperature dimension and can convert to each other. Thanks to the descriptive part of the schema, the related units in a dataset can be found. Units even can be present in a separate column in the database. The units and their mapped dimensions of attributes can be extracted and recorded separately. Table 2.3 shows that dimensions and units characterize some attributes of our running example. The dimension is also encoded using a one-hot encoding approach, as shown in Table 2.4.

Table 2.3 – *Some attributes with their units and associated dimension*

Attribute	Unit	Dimension
ENGINE_DISPLACEMENT	CCM	VOLUME
ENGINE_POWER	KW	POWER
PRICE_EUR	EUR	PRICE
ENGINE	CC	VOLUME

Table 2.4 – *One-hot encoding for converting dimensions to numerical feature*

Attribute	volume	power
ENGINE_DISPLACEMENT	1	0
ENGINE_POWER	0	1
ENGINE	1	0

Name Property The name of an attribute can also be useful for the analysis. Names often contain concatenated words and abbreviations. Thus, they first need to be normalized before they are used to construct a profile to compute linguistic similarities. As the first step, tokenization is applied, but it may not be enough;

e.g.,for the name 'vehicleType', the name should be split into the word 'vehicle' and 'Type'. In this regard, we compare all names of other attributes and see if one of them is part of the name string; this breakdown will be done.

Statistics Property The statistics profiles concern CATEGORICAL and NUMERICAL data types. PROCLAIM uses descriptive statistical analysis to produce a profile for each attribute which not only defines the characteristics of an attribute but also enables comparing the profiles to find similarities. We list the most important statistical measurements regarding NUMERICAL and CATEGORICAL data types in the following.

— NUMERICAL data type:

For the NUMERICAL data type, several measures can be studied. The domain under analysis and the analyzed data characteristics will help us select the significant ones. These measures can be variability or dispersion of the distribution of values per attribute, symmetry of the distribution, the number of instances (cardinality), and central tendency.

1. **Variability or dispersion** This metric defines a set's amount of data dispersion. The range, interquartile range, variance, and standard deviation are the four metrics that are used most frequently to describe or measure variability or dispersion.
2. **Symmetry of distribution** This measure shows how much a distribution of the values of an attribute differs from a normal distribution, either to the left or to the right side, and the skewness measure is used. This knowledge helps to know which measures of Variability are most important and where the majority of the values lie.
3. **Cardinality** This measure is the number of instances of an attribute and shows its importance. In Table 2.5, count represents the cardinality of the attribute.
4. **Central Tendency** This measure shows a central value for a probability distribution. The most popular measures for numerical variables are mean and median, but they can also be calculated through interquartile range and skewness.

— CATEGORICAL data type

For the CATEGORICAL data type, the considered statistics profile contains the top most frequent values among all instances of one attribute. This set of most frequent instances can design a pattern for an attribute.

Since other components of attribute profiles are encoded using a one-hot encoding approach, we decided to apply the same method to the statistics profile. First,

log transform will normalize the distribution with left or right skewness, then the distribution is presented into categorical scale using binning and finally encoded. We obviously lose the numerical nature of the statistics, but we can merge this vector easily with the other vectors without a normalization issue.

Table 2.5 – *Statistics Profile*

Attribute	5%	25%	50%	75%	95%	Count
DISP.	90.9	113.75	144.5	180.0	302.0	32
ENGINE	993.0	1198.00	1497.0	1995.0	2982.0	101
ENGINE_DISPLACEMENT	1124.0	1400.00	1600.0	1968.0	2967.0	158
ENGINE_POWER	44.0	65.00	80.0	103.0	161.5	114

Table 2.6 – *Normalized Statistics Profile*

Attribute	5%	25%	50%	75%	95%	Count
DISP.	5	5	5	5	6	3
ENGINE	7	7	7	8	8	5
ENGINE_DISPLACEMENT	7	7	7	8	8	5
ENGINE_POWER	4	4	4	5	5	5

In Table 2.5 we present the statistics profile for four numerical attributes. As a result of this analysis, we can see that the 'Engine' and 'Engine Displacement' have the same normalized distribution. Normalized data with log transformation is shown in Table 2.6.
For each attribute of the dataset, we compute the global profile, which is made of the four properties described in this section. Each profile is built by considering the attribute type, and the global profile is finally converted into a numerical vector.

We finally produce a collection of vectors that will be the input for the next computation steps.

We propose a weighting factor for each of the four properties that are adjusted according to the attribute's data type. For example, the attribute name can be ignored for numerical and categorical variables because this information is uncertain, and the distribution of the values is essential.

2.7 . Attribute Labeling

Attribute labeling is a three-step process that (1) performs attribute clustering, (2) assigns a label to each cluster, and (3) merges clusters having the same label. Step 3 creates every single attribute of the global schema. In this section, we will detail each step of the process.

2.7.1 . Clustering

The calibrated numerical vectors produced as described in Section 2.6 allow us to apply clustering to find similar groups of attributes ($C_i \in C$). PROCLAIM uses a density-based clustering method. Density-based clusters are connected, dense areas

in the data space separated from each other by low-density areas. Density-based clustering can be considered a non-parametric approach since this method makes no assumptions about the number or distribution of clusters [31]. In higher-dimensional space, the assumption of a certain number of clusters of a given distribution is very strong and may often be violated. However, other parameters should be identified, e.g., a density threshold that is the minimum number of points (MinPts) and the radius of a neighborhood (ϵ) in the case of DBSCAN [32] and OPTICS [33]. Sparse areas, as opposed to high-density areas, are considered outliers (noise), resulting in points in the sparse areas that are not assigned to any cluster since, in general, each outlier can be considered as one cluster containing just one element. As a result, 1) It is not necessary to specify the number of clusters; 2) not all the points need to belong to at least one cluster.

OPTICS (Ordering PoinTs to Identify the Clustering Structure) and DBSCAN are popular density-based clustering algorithms. Despite all the similarities in the core concept of both algorithms, they have fundamental differences [33]. However, One of DBSCAN's biggest weaknesses, the inability to identify meaningful clusters in data with variable densities, is addressed by OPTICS. For our purpose, because we encounter various densities for our variables, PROCLAIM uses OPTICS.

To better understand OPTICS, formal definitions for the notion of a density-based clustering as it was presented for the first time in [33] is shortly introduced in the following:

Directly density reachable: Directly density reachable: Let D be a set of numerical attribute profiles, p is a profile in this set ($p \in D$) and $N_\epsilon(p)$ is the ϵ-neighborhood of p. For each $x, y \in D$, profile y is directly density reachable from another profile x concerning (ϵ) and MinPts in D if:

1. y is in the ϵ-neighborhood x ($y \in N_\epsilon(x)$);
2. x is a core profile which means the Card($N_\epsilon(x)$) $\geq$ MinPts (Card(N) indicates the cardinality of the set N).

Density reachable: A profile x is density reachable from y concerning ϵ and MinPts in D if there is a chain of core profiles leading from y to x. In general, this relationship is not symmetric. Density reachable can only be made mutually with a core profile.

Density connected: Two profiles, y, and z, are density-connected profiles concerning ϵ and MinPts in D if there is a core profile x, such that both y and z are density reachable from x. Density connectivity is a symmetric relation.

Clusters and outliers: A cluster C_i ($C_i \in C$) concerning ϵ and MinPts in D is a non-empty subset of D satisfying the following conditions:

1. *Maximality*: $\forall x, y \in D$: if $x \in C$ and y is density reachable from x concerning ϵ and MinPts, then also $y \in C$.
2. *Connectivity*: $\forall x, y \in C$: x is density connected to y concerning ϵ and MinPts in D.

Figure 2.3 illustrates the definitions on a sample of a 2-dimensional profile.

Outlier or noise is defined as a set of profiles not belonging to any clusters $\{p \in D | p \notin C_i \forall_i\}$.

The method in DBSCAN defines a global density parameter (radius of a neighborhood ϵ) which is used as a threshold to define reachability. Different areas in the data space may have different densities, and it may not be possible to define clusters effectively with a global density parameter. OPTICS is an algorithm that produces a cluster ordering of the profiles concerning its density-based clustering structure. In principle, OPTICS works like an extended DBSCAN algorithm for an infinite number of distance parameters (ϵ) smaller than a generating distance. In principle, OPTICS works like an extended DBSCAN algorithm for an infinite number of distance parameters (ϵ) smaller than a generating distance. The only difference is that it does not assign cluster memberships but stores the order in which the profiles are processed (the clustering order) and the following two pieces of information, which an extended DBSCAN algorithm would use to assign cluster memberships.

Core distance of a profile p: Let MinPts be a natural number and let MinPts-distance(p) be the distance from p to its MinPts' neighbor. Then, the core distance of a profile p is the smallest distance between p and a profile in its ϵ-neighborhood such that p would be a core profile concerning this smallest distance if this neighbor is in $N_\epsilon(p)$ defined as:

$$= \begin{cases} Undefined & \text{if } Card(N_\epsilon(p)) < MinPts, \\ MinPts - distance(p) & \text{if } Otherwise. \end{cases} \tag{2.2}$$

Reachability distance of a profile p concerning profile o: The reachability distance of a profile p concerning profile o is the smallest distance which profile p is directly density reachable from profile o if o is a core profile and can be defined as:

$$= \begin{cases} Undefined & \text{if } |N_\epsilon(o)| < MinPts, \\ max(core - distance(o), distance(o,p)) & \text{if } Otherwise. \end{cases} \tag{2.3}$$

Density-based clusters pertaining to a higher density (e.g., a lower value for ϵ) are completely contained in clusters pertaining to a lower density (e.g., a higher value for ϵ) for a constant MinPts value. In PROCLAIM, we want to reduce the chain of the core profile effect in order to have small clusters with very similar profiles; hence, we set a very small value (e.g., 3) for the MinPts input of OPTICS. We will then compute many clusters and have many outliers. To reduce the number of outliers, we ran OPTICS a second time, again with a small value for the MinPts parameter only on the profiles that were considered as outliers. The clusters computed during the second step will be added to those computed in the first step. With these two iterations, as Extended OPTICS, we increase the number of clusters and reduce the outliers.

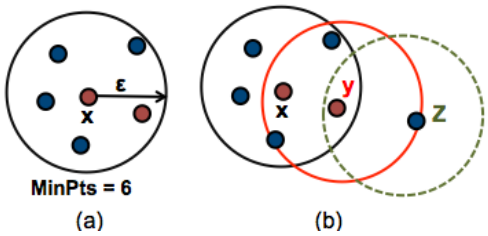


Figure 2.3 – *Principles of density-based clustering: x is a core profile, y is a part of the cluster but is not a core profile, and Z is an outlier. y is directly density reachable from x [33]*

2.7.2 . Extended OPTICS

In PROCLAIM, we propose an extended version of OPTICS by introducing a dynamic minimum number of points (MinPts) in place of a static one. A generative radius of neighbors to have different minimum numbers of points (MinPts) for clustering the unassigned points (outliers) derived from the OPTICS method.

The outliers, mostly the points from the low-density area of data space, can be assigned to new clusters with a new value of MinPts. Let $D\prime$ be a set of outlier points not belonging to any clusters distinguished by OPTICS concerning an initial amount of Minpts and generative ϵ, $\{p_o \in D\prime | p_o \notin C_i \forall_i, D\prime \subset D$ where p_o is a profile and D is an initial data space$\}$. By decreasing the MinPts value by one and applying OPTICS again on the new data space ($D\prime$), we will compute new clusters concerning a generative radius and new MinPts value, resulting in the discovery of new clusters. Then, the process can be applied again until MinPts reaches the minimum value that can define a cluster (2). For attribute matching, we expect that there should be a large number of clusters with just two elements. The goal is to define the clusters at their finest level.

With this method, we increase the final number of clusters produced from the initial data space by OPTICS. However, with such a low MinPts value, there is the possibility of dividing an inclusive cluster of points into two or more subclusters, even if they refer to the same attribute. PROCLAIM provides a solution for this problem by merging similar clusters. The merging process is defined as a *Labeling Function* that merges clusters labeled by the same labels (automatically assigned to clusters) in detail, explained in the following section.

2.7.3 . Labeling Function

The labels for each cluster will be created using the descriptions and names of all elements in each cluster. The stop words will be removed using the common linguistic and domain-specific words. The idea is to select the most frequent words, bigram, and trigram terms appearing in the description and name of each attribute of the cluster. Then, the most frequent term will be the cluster's label, as shown in Example 3.

Example 3. *Consider* $C_1 = \{Engine, Disp.\}$ *as a cluster computing using the two-steps OPTICS algorithm. The descriptions gathered per each attribute are:*
$Descr\_Engine$ = *' The displacement volume of the engine in CC.'*
$Descr\_Disp.$ = *': Represents the engine displacement of the car'*
The Name profile of attributes can also be added to the descriptions: Descr_names = $\{engine, disp\}$.
Furthermore, after removing the stop words, the following bag of words for each description will be generated:
BOW_Engine = {displacement : 1, *volume* : 1, *engine* : 1, *cc* : 1$\}$
BOW_Disp. = {represents : 1, *engine* : 1, *displacement* : 1, *car* : 1$\}$
BOW_names. = {engine : 1, *disp* : 1$\}$
Moreover, we create a holistic bag of words by merging all the terms associated with their total number of occurrences as follows:
BOW_total = {engine : 3, *displacement* : 2, *volume* : 1, *cc* : 1, *represents* : 1$\}$

By selecting the most represented term, we may produce meaningless labels such as "displacement engine" rather than "engine displacement". To tackle this problem, we need to create a domain-specific corpus and extract from it the bigrams and trigrams associated with the respective number of occurrences, which will result in adjusting and validating the labels.

Consider a created corpus in the car domain, which includes resources of glossaries, dictionaries, and wikis, which can easily be gathered online [1]. Now, all combinations of the highest frequency words from BOW_total will be considered to create the bigrams and trigrams which already exist in this domain (the meaningful N-grams) concerning terms' frequency in the corpus. The bigrams and trigrams will create a valid bag of terms. We will also add the most frequent word appearing in the corpus to this valid bag of terms. From Example 3, we have: $Bag\_of\_terms =$ {engine displacement : 2, displacement volume : 1, engine : 3$\}$. To get the selected label, we take from the bag of terms the term with the maximum number of occurrences with priority first to the trigrams, then bigrams, and finally, words.
The selected label for the cluster $C_1 = \{ENGINE, DISP.\}$ is ***engine displacement*** even if the number of occurrences of word ***engine*** is higher.

After labeling each cluster, we can finally merge the clusters with the same label or labels that are synonyms (Example 4).

1. Data from https://www.kaggle.com/

Example 4. *Consider C_2=$\{ENGINE\_DISPLACEMENT, ENGINE\_POWER\}$ as another cluster computed using the two-step OPTICS algorithm. The bag of words retrieved from related descriptions for these attributes are:*
$BOW\_Engine\_Displacement$ = {*ccm* : 1}
BOW_Engine_Power = {*kw* : 1}
BOW_names = {*engine* : 2, *displacement* : 1, *power* : 1}
As the final result, the output is:
Bag_of_terms = {*engine displacement* : 2, *engine power* : 1, *engine* : 3}

*The computed label is **engine displacement** which means that this cluster can be merged with cluster C_1 of example 3. Then the new cluster contains the following attribute {ENGINE, DISP., ENGINE_DISPLACEMENT, ENGINE_POWER}.*

All the merged and labelled clusters generate a global schema for a specific domain. The label of different clusters in different data types could be the same, which enables us to integrate the attributes together even if their data types were assigned wrongly in Section 2.5. PROCLAIM helps to integrate the data from different sources and creates a general schema that can help integrate new sources or populate a knowledge base in a specific domain.

2.7.4 . PROCLAIM Global Schema

As explained in the Labeling Function process in Section 2.7.3, the final goal of PROCLAIM is to automatically provide a set of labels to represent a specific domain for given different schemas, and it provides the clusters of matched attributes from thoseschemas. In our approach, the set of vocabularies is the global target schema of the domain. For example, the set of vocabularies (global schema) for the car domain retrieved from the Car_Kaggle dataset is shown in Example 5.

Example 5. *The output labels grouped according to their data type are:*
L_Numerical = {federal information, power, weight, engine displacement, price converted, kilometers, year, vehicle latitude}
L_Categorical = {name slug, fuel type, automatic transmission, count, listing, state code}
L_Date = {last date}

Then it will lead to having a final set of labels as the global schema for this dataset as follows:

Global Schema = {federal information, power, weight, engine displacement, price converted, kilometers, year, vehicle latitude, name slug, fuel type, automatic transmission, count, listing, state code, last date}.

2.8 . Experiments and Analysis

In this section, we provide the experimental results on two datasets: one of them is our running example about cars, and the second is from the oil and gas domain.

2.8.1 . Environment

The code of the experiment is implemented in Python 3.6.7. Parquet [34] is used to store original datasets. Parquet is a column-oriented data storage that is free, open-source, optimized, and developed on the Apache Hadoop platform.

2.8.2 . Datasets

To our knowledge, there are no benchmark-labeled datasets for comparing our results with another method. Therefore, we have collected data from Kaggle challenges for the car example. For the Oil and Gas example, we use a large dataset. We look at the Oil_NorthSea datasets in detail, and the data preprocessing steps are done to prepare the data. The same process has been done for the Car_Kaggle dataset, but since the number of datasets is small and we used these datasets as an example, we directly present the experiment results in Section 2.8.3.

Car_Kaggledataset The Car_Kaggle dataset was gathered from five different sources (S_1, ..., S_5) about cars from different Kaggle challenges[1]. The global Car_Kaggle dataset, which merges all sources, contains 78 original attributes.

Oil_NorthSea dataset The North Sea Oil and Gas (Oil_NorthSea) dataset was gathered from OGA [2] (The Oil and Gas Authority Open Data) website, which contains 43,997 different sources with a total of 5260 attributes assigned to 12583 unique well IDs. Based on previous knowledge at the company, we know we have 12183 well in the north sea. well_ids can follow different standards.

Figure 2.4 – *Distribution of locations of wells in the (Oil_NorthSea) dataset*

The distribution of the location of the good origins of wells is presented in Figure 2.4. A few of them have the locations except than oil north sea area, which can indicate two reasons: (1) the latitude and longitude for those wells are

2. The data is published under Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0).

not provided correctly in the primary dataset; (2) the latitude and longitude are correct, and those wells do not belong to the targeted area and should be filtered. However, except for these few wells, most are in the targeted area, as shown in Figure 2.5.

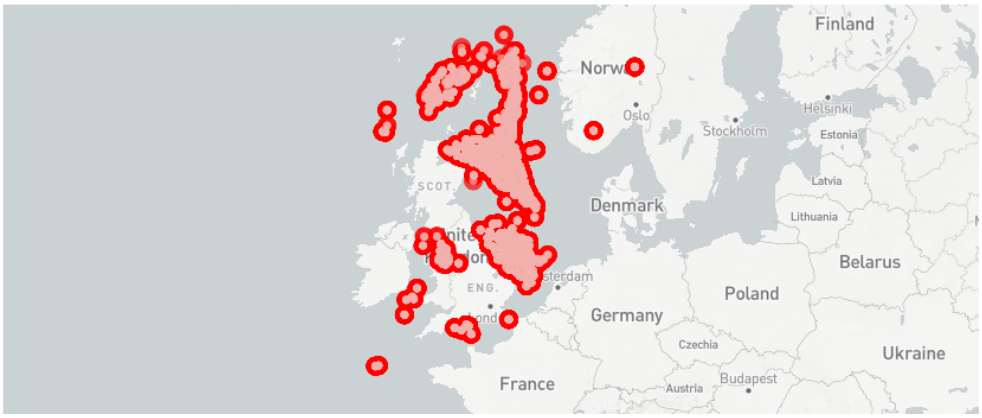


Figure 2.5 – *Distribution of locations of wells in the North Sea region*

Looking at the data, we can explore interesting information about these wellbores, such as in Figures 2.7 and 2.8.

After applying the data integration step, these datasets were gathered in a columnar format in Parquet. This integrated dataset is too large, and data cleaning is very time-consuming and expensive, however as mentioned in section 2.4.2, some steps are necessary to be applied:

— **Harmonizing Data Values** Attributes that contain 'date' in their names were chosen (for sure, there are more attributes that contain date values but finding them automatically in this step was not the goal). Fifty-two parameters were found; the date format changed to this format: Day-Abbreviation of Month-Year; for example, '12-APR-2001'.

— **Unique well IDs Detection** The problem with unique "well IDs" is that different sources follow different standards for naming the same well. Then the same well can be represented with different names in different sources. We know that we have 12183 unique wells, in reality, [3] but in our dataset, there are 12853 unique well IDs; after harmonizing the form of well IDs, the dates related to each well were compared with each other, and a small labeled dataset was created representing which name of wells are equal and which ones are not (see Figure 2.6). We kept one specific format for all equal wells and finally reached 12212 unique wells. The number is not precisely 12183, but around 640 wells could be matched to their correspondence wells.

3. (This information was provided to us from Equinor company - One of the main Schlumberger customers in the North Sea

	Wbi1	Wbi2	Label
0	1-4-2001	1-04-1	1.0
1	1-4-2002	1-04-2	1.0
2	10-1-2002	10-01-2	1.0
3	10-1-2003	10-01-3	1.0
4	10-1-2004	10-01-4	1.0

Figure 2.6 – *a sample of unique well IDs labeled dataset*

— **Handling Missing Values** Deleting the Non-value data in Parquet format is so easy.

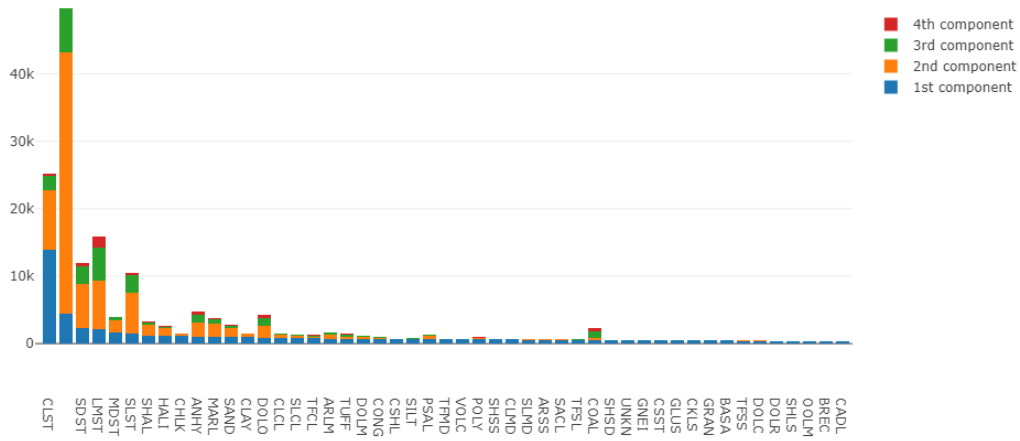


Figure 2.7 – *This chart shows the frequency of lithology terms when the rock in each depth is cut. The order is based on the frequency of the first component*

2.8.3 . Evaluation and Analysis

Table 2.7 – *Car_Kaggle dataset Information as the input for PROCLAIM*

Dataset	Kaggle Challenge Name	#Attributes	#Descriptions	#Units	#Source records
S_1	Used Cars Price Prediction	13	11	4	1000
S_2	cars data	8	7	0	600
S_3	personal cars classified	16	11	4	1000
S_4	Craigslist Cars EDA	26	24	0	1000
S_5	Used cars database	20	12	1	1000
Sum	Car_Kaggle	70	65	9	4600

Figure 2.8 – *Distribution of different lithology elements per each well in different depths.*

Car_Kaggle As we mentioned in Section 2.8.2, the Car_Kaggle dataset contains 78 original attributes: 70 have different names; 65 out of 70 attributes contain descriptions, and just 9 out of 70 attributes have the provided unit. In Table 2.7, we provide the details of each schema. As the first step, we run data type identification to discover each attribute's type. Data for this dataset can be split into four different types; as we can see, the RARE data type is not present. Unique attributes are discarded (6 attributes), and we compute the profile for the 64 remaining attributes (25 NUMERICAL, 35 CATEGORICAL, and four DATE attributes) and automatically assign a label to each attribute. To be able to evaluate PROCLAIM, we manually labeled all the attributes. A subset of PROCLAIM and manual labels can be seen in Table 2.10.

Table 2.8 – *number of attributes with different data types*

Dataset	#Numerical Attr. *	#Categorical Attr.	#Date Attr.	#Rare Attr.	#Unique Attr.
Car_Kaggle	25	35	4	0	6

The number of labels for different data type attribute sets is given in Table 2.9.

We used three metrics to evaluate the quality of PROCLAIM labels: *precision*, *recall*, and *F-measure*. Precision is defined as the percentage of the correct labels. We compared manual labels with PROCLAIM labels. If the pair (PROCLAIM LA-

Table 2.9 – *Number of automatic labels as the final result of PROCLAIM*

Data type	#Generated Labels	#attributes with label	#attributes without label
Numerical	8	25	10
Categorical	6	15	20
Date	1	4	0

Table 2.10 – *Number of automatic labels as final result of PROCLAIM*

Attributes	PROCLAIM labels	Annotated labels	Match
PRICE_EUR	price converted	price	1
PRICE	price converted	price	1
POWERPS	power	power	1
HP	power	power	1
WEIGHT	weight	weight	1
POSTALCODE	weight	address	0

Table 2.11 – *Number of automatic labels as the final result of PROCLAIM*

Data type	Precision	Recall	F-measure
numerical	85.7	85.7	85.7
categorical	73.0	58.8	64.2
date	100	100	1
Overall	82.5	72.7	77.3

BEL, MANUALLY ANNOTATED LABEL) matches, the label is valid, as seen in 2.10. The recall is the ratio of attributes with correct labels to all attributes (with or without labels). F-measure is the harmonic mean of precision and recall. This result is shown in Table 2.11. These measures were calculated separately for each set of attributes (of each data type) and finally for the whole attributes. As can be seen in Table 2.11, precision is showing a good quality of labels, but since the number of attributes and sources is not considerable, we expected not very high recall, but still, this recall is promising for the schema matching problem, which in this research is not the primary concern. The main goal is to have labels of high quality.

Table 2.12 – *PROCLAIM Evaluation*

Data type	Precision	Recall	F-measure
numerical	85.7	85.7	85.7
categorical	73.0	58.8	64.2
date	100	100	1
Overall	82.5	72.7	77.3

Table 2.13 – *Oil North-Sea Datasets Information*

Dataset	#Sources	#Attributes	#Columnar format records	#Descriptions	#Units
Oil_NorthSea	43997	5260	2713222	3481	1668

Oil_NorthSea dataset As mentioned in Section 2.8.2, the North Sea Oil and Gas (Oil_NorthSea) dataset contains 43,997 different sources with a total of

5,260 attributes, 4,713 of which have different names. The description is available for 3,481 attributes, and the unit is provided for 1,668 over 4,713 attributes. We apply the same approach as described in Section 2.8.3. The number of different identified types of attributes is 638 NUMERICAL, 631 CATEGORICAL, 46 DATE, 574 RARE, and 2,824 UNIQUE attributes. Since the number of attributes is too big to be manually annotated, we asked domain experts to label a random set of attributes (20 labels for NUMERICAL and CATEGORICAL attributes and all labels for DATE attributes - the number of DATE type attributes are less than 50). We cannot calculate recall and f-measure here since the manual labels are just provided for a subset of random labels. However, precision is calculated for these subsets for different experiments. Experiments are done for different profiles for each group of the same data type attributes, and the result is shown in Table 2.14. The cover data ratio measures the percentage of labeled attributes. The coverage ratio shows a high percentage of considered attributes to discover the global schema. As can be seen, the precision of clusters for NUMERICAL, CATEGORICAL, and DATE data type is over 90%, which is a promising result. The global schema created from the Oil_NorthSea dataset contains 247 labeled attributes which cover 86% of the 1,315 original attributes belonging to the NUMERICAL, CATEGORICAL, and DATE data types.

Table 2.14 – *Experiments results for different profiles subset*

Data Type	Profile	Unlabeled Attribute	Labeled Attribute	Labels	Precision (%)	Coverage (%)
Numerical	Stat.	84	554	107	58.1	86.8
	Descr.	122	516	112	93.25	80.9
	[Stat., Descr.]	53	585	128	86.4	91.6
	[Stat.,Descr.,Unit]	60	578	110	90.1	90.5
Categorical	Stat.	135	496	102	70.5	78.6
	Descr.	203	428	100	94.9	67.8
	[Stat., Descr.]	129	502	126	86.2	79.5
	[Stat.,Descr.,Unit]	121	510	130	92.1	80.8
Date	Descr.	16	30	3	100	65.2
	[Descr.,Name]	5	41	7	94.3	89.1
	[Descr.,Unit,Name [1]]	5	41	7	94.3	89.1
Total	**[full profile]**	**186**	**1129**	**247**	**92.2**	**85.9**

[1]Unit is not available for Date attributes

2.9 . Conclusion

Compared to the massive work on pairwise schema matching, research on holistic schema matching for more than two sources is still at an early stage. PROCLAIM is an efficient and effective schema-matching method and provides a consistent domain-specific attribute schema. Experiments show that thanks to our approach, we can automatically gather more than 80% of the vocabulary related to a domain and populate the knowledge bases with corresponding attributes from heterogeneous sources. In future work, our approach can be extended for handling new attributes from new sources and enriching the labels by adding similar words from different thesauri and dictionaries. PROCLAIM can provide an efficient and effective way for schema matching. Also, this method can provide a global attribute schema in a specific domain. In the real world, we often lack vocabulary as the starting point for populating domain-specific knowledge bases. Even for some processes, such as automatic knowledge base construction, manual effort, in the beginning, is an essential part. However, with PROCLAIM, we can automatically gather more than 80% of the vocabulary related to a domain and populate the knowledge bases with corresponding attributes from heterogeneous sources. On the other hand, we still encounter the cold start problem for completely new attributes from new sources. To solve this problem, one way is using the resultant global schema from existing tools such as Flexmatcher [16]. These existing tools are based on pairwise matches, which need a mediated schema. However, PROCLAIM can be used in several domains to create the basis for an automatic knowledge base construction regarding many data sources in one domain. Although the labels are provided by PROCLAIM automatically with good quality, these labels can be enriched by adding similar words from different thesauri and dictionaries. Also, an additional step to properly handle abbreviations existing in descriptions could be added to increase the quality of the labels.

The main goal of PROCLAIM is discovering the global attribute schema in a domain which is covering the most attributes. However, this method has its own limitations. Sometimes, there are not enough statistical or contextual features for each attribute to distinguish different attributes completely. Also, based on the PROCLAIM method, two attributes with the same statistical distribution, datatype, similar names, and context may look likely to be the same. However, this is not always the case; e.g., *Company_Owner_name* and *Company_drilling_name* are two different attributes. However, both may be assigned to the same label (e.g., *Company*) by PROCLAIM. In future research, one solution to this challenge can be finding the sub-labels based on new criteria for attributes assigned to the same label. In this case, we will have hierarchy levels of attributes and more detailed labels, which can increase the precision and create a better global schema. Besides the central research challenge of this chapter, PROCLAIM can create large labeled datasets. The quality of the labels is an important question that can only be evaluated sometimes by asking an expert to do the manual evaluation. Weak Supervision

is a promising method here. A denoising Framework like Snorkel [35] can potentially boost accuracy over PROCLAIM method as labeling functions (LFs) for the Framework. For some categorization tasks, Snuba-on-snorkel [36] can automate the generation of LFs. The fact that Snorkel does not now accept complicated prediction outputs like PROCLAIM labels to consider them as LFs directly is an interesting research challenge.

3 - Machine Learning for Document Structure Recognition

3.1 . Introduction

Hundreds of thousands of reports and PDF files are produced worldwide every day. Some businesses or governmental organizations need to process them fast. Sometimes this data contains the core data of the business, and sometimes it helps to enrich the analysis of the industry. Most companies must transfer data from these PFDs and forms into digital databases. Many companies extract data from internal and external documents through manual efforts, which is time-consuming and expensive. These documents are not only of different structures, but they may also be digital-born such as PDF files, or in a scanned form that comes from the handwritten or printed papers. Therefore, it is necessary to process them while keeping the initial structure to make them understandable by the computers and extract relevant information.

The development of optical character recognition (OCR) systems for automatic text reading has received a lot of scientific focus in recent years [37] [38] [39] [40]. OCR format has been widely used due to the need for mass digitization of historical documents. Various formats for representing the final results of OCR engines have been proposed. These ad-hoc formats are often developed to satisfy the practical requirements of the applications. Much effort has been invested in the layout analysis of PDF documents and images. [41] [42] [43] [44] [45]. The layout structures have been classified as physical (text, tables, pictures, charts, graphics, ...) or logical (title, author, abstract, header, footer, captions, ...). The layout analysis can be considered the first step for advanced analytics and the heart of information extraction from unstructured data.
During the last decades, many geological reports have accumulated in the Oil and Gas industry during exploration, development, production, and other procedures for wells. Schlumberger company had the opportunity, thanks to its business, to collect, produce and store these geological reports on different geological themes: geological surveys, cutting reports, end of well reports, and core lab reports; and talk about separate analyses such as the geological analysis, the study of material extended from the mud during the drilling process, the formation and similar information when the well is drilled, or the core extraction analysis of the formations in the laboratory. These reports - PDF files or images - differ since they are produced by different tools, following conventions of other countries, and performed by heterogeneous experts during different timelines.

Some samples of pages of these heterogeneous structured documents are shown in Figures 3.1, 3.2, 3.3, and 3.4. As shown in Figure 3.1, we have different tables;

some pages include one specific format of papers (Figures a, b in 3.1), while others can consist of two or more different table formats (Figures c, d in 3.1). Figure 3.2 shows some pages, including graphical contents and figures. The text content in both tables and figures is a *Non-Dense* part of PDF files since the text content is not dense. We call these pages *Non-Dense* because their structures contain the Non-Dense part of the text. All figures in 3.1 and 3.2 are *Non-Dense*. However, Figure 3.3 shows examples of *Dense* text pieces in paragraphs.

Some scanned images are from handwritten geological reports (Figure 3.4 b); some are from geological reports with handwritten marks on them (Figure 3.4 c, d); and some are from reports with stamp marks on them (Figure 3.4 a). This kind of remark decreases text quality and makes its content more difficult to exploit.

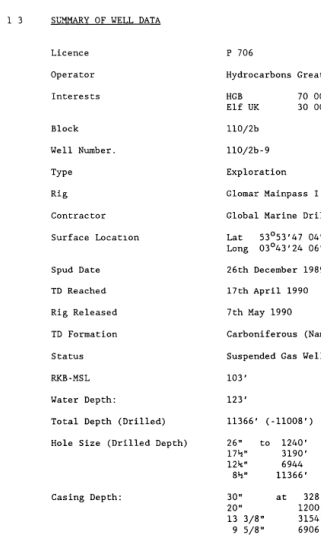
To understand the content of these documents, we first need to extract the text from images while keeping the same structure of documents to proceed with other advanced analytics. If we consider the structure of a document as a hierarchy scheme, from a coarse-grained document element level to finer element ones such as paragraphs, lines, tokens, and characters, we would like to have a fine-grained annotation scheme. We can store the related information regarding a character. Still, since tokens (words) are the finest element level of a document semantically for humans, in our hierarchy scheme, we consider the tokens as the finest element level to be stored. This hierarchy scheme is explained in detail in 3.4.1.

The main issues which we encounter with these documents when we try to get the text as the output of OCR engines are:

— The output of most OCR engines provides text pages that mainly include the bounding boxes of large semantic structures. The OCR engines mainly provide the bounding boxes of the fine-grained elements in a document hierarchy (Characters). However, to keep the same structure of the documents, we need fine-grained element-level scheme annotations. These elements can form the structure of the files, such as characters, tokens, lines, etc. There is a necessity for large-scale datasets in a machine-understandable format for all documents.

— The output of most OCR engines due to the strike-through text or low quality of scanned images has a low confident rate for specific characters (as shown in Figure 3.4). Cleaning and correcting the original documents are time-consuming tasks. Therefore, we have also to deal with the low quality of OCR outputs.

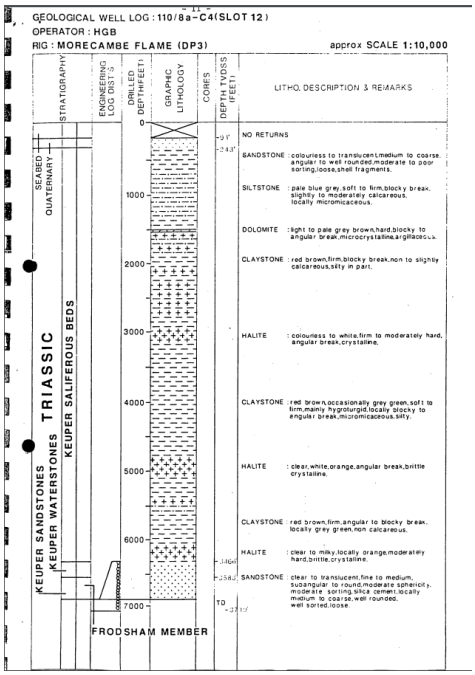
When we have the document context in the fine-grained token-level scheme database, we must distinguish each line's layout label. As we mentioned, in their format, documents and PDF files have two kinds of *Dense* and *Non-Dense* pieces. The *Dense* part of the text, such as paragraphs, and the *Non-Dense* components, such as tables and figures. If we have a look into our fine-grained line-level scheme of the database of our reports, we would like to classify each line of documents in one of the two categories: *Dense*, which means this line is part of paragraphs of



1 3 SUMMARY OF WELL DATA

Licence	P 706
Operator	Hydrocarbons Great Britain Limited
Interests	HGB 70 00% Elf UK 30 00%
Block	110/2b
Well Number.	110/2b-9
Type	Exploration
Rig	Glomar Mainpass I
Contractor	Global Marine Drilling Company
Surface Location	Lat 53°53'47 04"N Long 03°43'24 06"W
Spud Date	26th December 1989
TD Reached	17th April 1990
Rig Released	7th May 1990
TD Formation	Carboniferous (Namurian A-B)
Status	Suspended Gas Well
RKB-MSL	103'
Water Depth:	123'
Total Depth (Drilled)	11366' (-11008')
Hole Size (Drilled Depth)	26" to 1240' 17½" 3190' 12¼" 6944 8½" 11366'
Casing Depth:	30" at 328' 20" 1200' 13 3/8" 3154' 9 5/8" 6906'

(a) *Page including one table format*



(b) *Page including one table format*

CONTENTS

(c) *Page including two tables formats*

Missed Samples due to excessively fast drilling

930ft	1170ft	1740ft	2370ft	2400ft
2800ft	2820ft	2830ft	2840ft	2850ft
2860ft	2870ft	2880ft	2890ft	2900ft
2920ft	2930ft	2940ft	2950ft	2970ft
2980ft	3000ft	3020ft	3030ft	3050ft
3070ft	3080ft	3100ft	3110ft	3120ft
3130ft	3150ft	3160ft	3190ft	3210ft
3220ft	3240ft	3250ft	3270ft	3290ft
3310ft	3320ft	3330ft	3350ft	3370ft
3390ft	3400ft	3420ft	3430ft	3450ft
3470ft	3480ft	3500ft	3510ft	3530ft

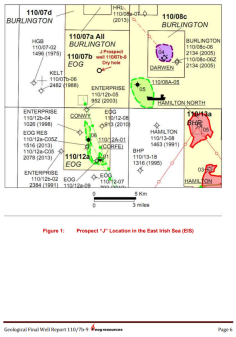
RockWash Prep & Store Ltd were contracted to prepare and photograph a small proportion of the washed/dried set prior to forwarding to ALS Petrophysics Ltd in Normandy (SURREY) for storage. The **Rock**Wash boxed end product can be found at EOG UKL's Guildford office.

Samples are stored at the below addresses:

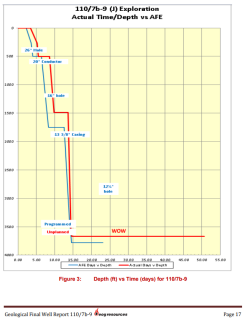
Set	Type	Purpose	Distribution
A	Unwashed wet for **Rock**Wash and Henley Park storage	EOG	**Rock**Wash Prep & Store Ltd., Unit 23, Evan Business Centre, Western Ind. Est., Caerphilly, South Wales. CF83 1BE Were then forwarded to: ALS Petrophysics Ltd., Unit 1A / 1B Henley Park, Normandy, Guildford, SURREY. GU3 2DX
B	Washed / dried	DECC	Scott Renshaw, Core Facilities Manager, National Geological Repository, BGS, Keyworth, Nottingham. NG12 5GG
C	Washed / dried	EOG	Kirk Petrophysics Ltd., Unit 1A / 1B Henley Park, Normandy, Guildford, SURREY. GU3 2DX

(d) *Page including two tables formats*

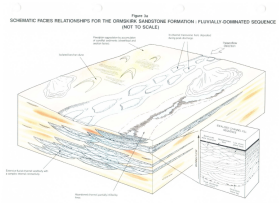
Figure 3.1 – *Samples of scanned images of geological reports with different layouts and format- different table formats*



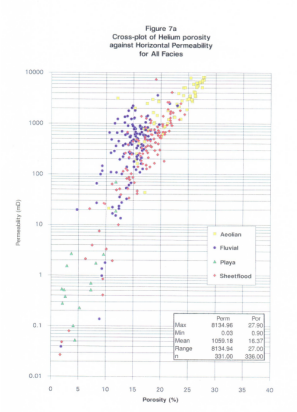
(a)



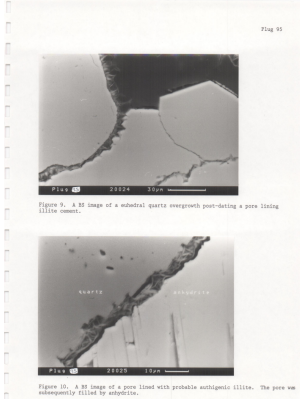
(b)



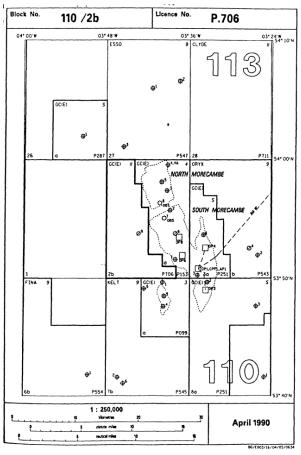
(c)



(d)

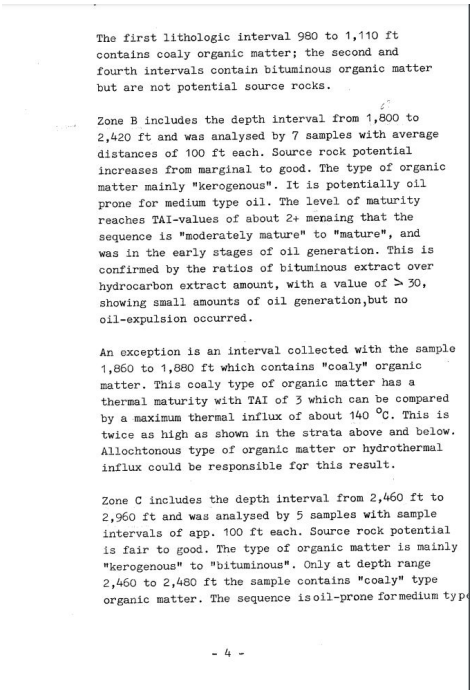


(e)



(f)

Figure 3.2 – *Samples of scanned images of geological reports with different layouts and format- different figure formats*



The first lithologic interval 980 to 1,110 ft contains coaly organic matter; the second and fourth intervals contain bituminous organic matter but are not potential source rocks.

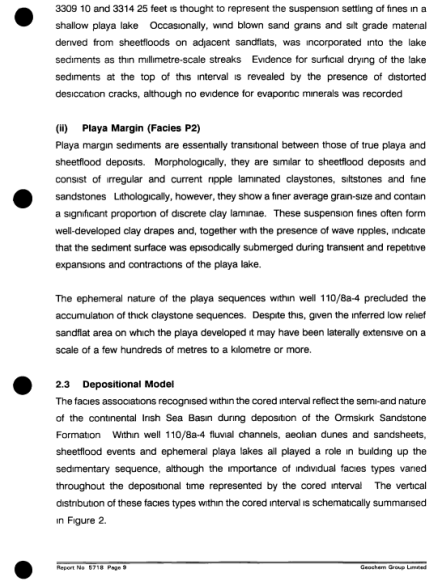
Zone B includes the depth interval from 1,800 to 2,420 ft and was analysed by 7 samples with average distances of 100 ft each. Source rock potential increases from marginal to good. The type of organic matter mainly "kerogenous". It is potentially oil prone for medium type oil. The level of maturity reaches TAI-values of about 2+ menaing that the sequence is "moderately mature" to "mature", and was in the early stages of oil generation. This is confirmed by the ratios of bituminous extract over hydrocarbon extract amount, with a value of > 30, showing small amounts of oil generation,but no oil-expulsion occurred.

An exception is an interval collected with the sample 1,860 to 1,880 ft which contains "coaly" organic matter. This coaly type of organic matter has a thermal maturity with TAI of 3 which can be compared by a maximum thermal influx of about 140 °C. This is twice as high as shown in the strata above and below. Allochtonous type of organic matter or hydrothermal influx could be responsible for this result.

Zone C includes the depth interval from 2,460 ft to 2,960 ft and was analysed by 5 samples with sample intervals of app. 100 ft each. Source rock potential is fair to good. The type of organic matter is mainly "kerogenous" to "bituminous". Only at depth range 2,460 to 2,480 ft the sample contains "coaly" type organic matter. The sequence is oil-prone for medium ty

- 4 -

(a) *Page including four paragraphs*



3309 10 and 3314 25 feet is thought to represent the suspension settling of fines in a shallow playa lake Occasionally, wind blown sand grains and silt grade material derived from sheetfloods on adjacent sandflats, was incorporated into the lake sediments as thin millimetre-scale streaks Evidence for surficial drying of the lake sediments at the top of this interval is revealed by the presence of distorted desiccation cracks, although no evidence for evaporitic minerals was recorded

(ii) Playa Margin (Facies P2)

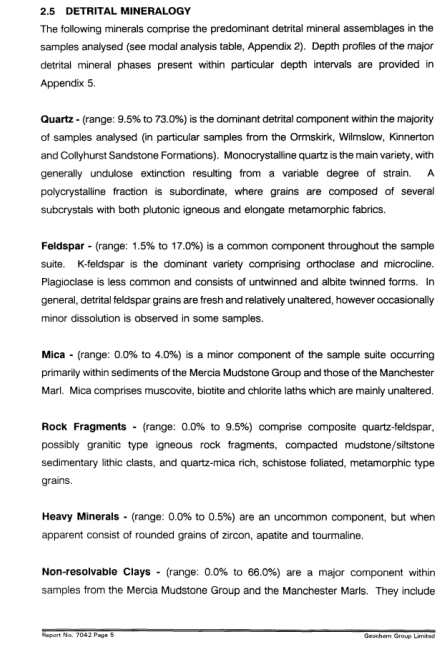
Playa margin sediments are essentially transitional between those of true playa and sheetflood deposits. Morphologically, they are similar to sheetflood deposits and consist of irregular and current ripple laminated claystones, siltstones and fine sandstones Lithologically, however, they show a finer average grain-size and contain a significant proportion of discrete clay laminae. These suspension fines often form well-developed clay drapes and, together with the presence of wave ripples, indicate that the sediment surface was episodically submerged during transient and repetitive expansions and contractions of the playa lake.

The ephemeral nature of the playa sequences within well 110/8a-4 precluded the accumulation of thick claystone sequences. Despite this, given the inferred low relief sandflat area on which the playa developed it may have been laterally extensive on a scale of a few hundreds of metres to a kilometre or more.

2.3 Depositional Model

The facies associations recognised within the cored interval reflect the semi-arid nature of the continental Irish Sea Basin during deposition of the Ormskirk Sandstone Formation Within well 110/8a-4 fluvial channels, aeolian dunes and sandsheets, sheetflood events and ephemeral playa lakes all played a role in building up the sedimentary sequence, although the importance of individual facies types varied throughout the depositional time represented by the cored interval The vertical distribution of these facies types within the cored interval is schematically summarised in Figure 2.

(b) *Page including four paragraphs*



2.5 DETRITAL MINERALOGY

The following minerals comprise the predominant detrital mineral assemblages in the samples analysed (see modal analysis table, Appendix 2). Depth profiles of the major detrital mineral phases present within particular depth intervals are provided in Appendix 5.

Quartz - (range: 9.5% to 73.0%) is the dominant detrital component within the majority of samples analysed (in particular samples from the Ormskirk, Wilmslow, Kinnerton and Collyhurst Sandstone Formations). Monocrystalline quartz is the main variety, with generally undulose extinction resulting from a variable degree of strain. A polycrystalline fraction is subordinate, where grains are composed of several subcrystals with both plutonic igneous and elongate metamorphic fabrics.

Feldspar - (range: 1.5% to 17.0%) is a common component throughout the sample suite. K-feldspar is the dominant variety comprising orthoclase and microcline. Plagioclase is less common and consists of untwinned and albite twinned forms. In general, detrital feldspar grains are fresh and relatively unaltered, however occasionally minor dissolution is observed in some samples.

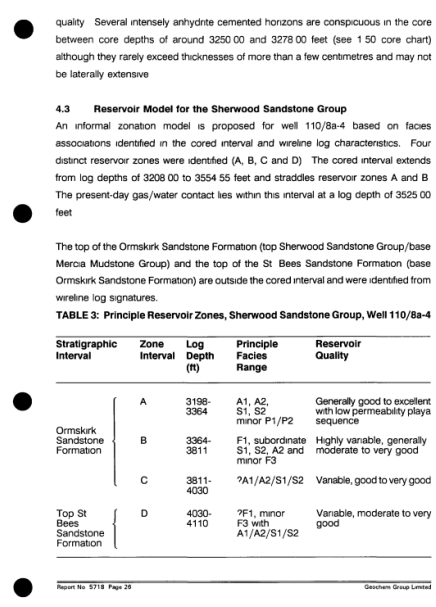
Mica - (range: 0.0% to 4.0%) is a minor component of the sample suite occurring primarily within sediments of the Mercia Mudstone Group and those of the Manchester Marl. Mica comprises muscovite, biotite and chlorite laths which are mainly unaltered.

Rock Fragments - (range: 0.0% to 9.5%) comprise composite quartz-feldspar, possibly granitic type igneous rock fragments, compacted mudstone/siltstone sedimentary lithic clasts, and quartz-mica rich, schistose foliated, metamorphic type grains.

Heavy Minerals - (range: 0.0% to 0.5%) are an uncommon component, but when apparent consist of rounded grains of zircon, apatite and tourmaline.

Non-resolvable Clays - (range: 0.0% to 66.0%) are a major component within samples from the Mercia Mudstone Group and the Manchester Marls. They include

(c) *Page including seven paragraphs*



quality Several intensely anhydrite cemented horizons are conspicuous in the core between core depths of around 3250 00 and 3278 00 feet (see 1 50 core chart) although they rarely exceed thicknesses of more than a few centimetres and may not be laterally extensive

4.3 Reservoir Model for the Sherwood Sandstone Group

An informal zonation model is proposed for well 110/8a-4 based on facies associations identified in the cored interval and wireline log characteristics. Four distinct reservoir zones were identified (A, B, C and D) The cored interval extends from log depths of 3208 00 to 3554 55 feet and straddles reservoir zones A and B The present-day gas/water contact lies within this interval at a log depth of 3525 00 feet

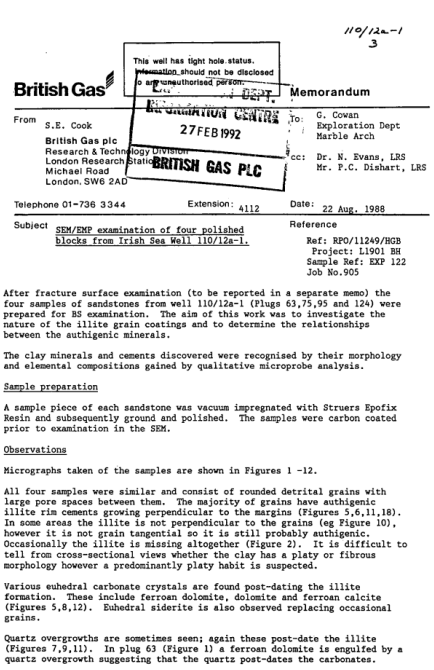
The top of the Ormskirk Sandstone Formation (top Sherwood Sandstone Group/base Mercia Mudstone Group) and the top of the St Bees Sandstone Formation (base Ormskirk Sandstone Formation) are outside the cored interval and were identified from wireline log signatures.

TABLE 3: Principle Reservoir Zones, Sherwood Sandstone Group, Well 110/8a-4

Stratigraphic Interval	Zone Interval	Log Depth (ft)	Principle Facies Range	Reservoir Quality
Ormskirk Sandstone Formation	A	3198-3364	A1, A2, S1, S2 minor P1/P2	Generally good to excellent with low permeability playa sequence
	B	3364-3811	F1, subordinate S1, S2, A2 and minor F3	Highly variable, generally moderate to very good
	C	3811-4030	?A1/A2/S1/S2	Variable, good to very good
Top St Bees Sandstone Formation	D	4030-4110	?F1, minor F3 with A1/A2/S1/S2	Variable, moderate to very good

(d) *Page including three paragraphs and one table*

Figure 3.3 – *Samples of scanned images of geological reports with different layouts and format-containing dense parts of the text (Paragraphs)*



110/12a-1
3

This well has tight hole status. Information should not be disclosed to any unauthorised person.

BRITISH GAS PLC

British Gas

Memorandum

From S.E. Cook
British Gas plc
Research & Technology Division
London Research Station
Michael Road
London. SW6 2AD

To: G. Cowan
Exploration Dept
Marble Arch

cc: Dr. N. Evans, LRS
Mr. F.C. Dishart, LRS

Telephone 01-736 3344 Extension: 4112 Date: 22 Aug. 1988

Subject SEM/EMP examination of four polished blocks from Irish Sea Well 110/12a-1.

Reference
Ref: RPO/11249/HGB
Project: L1901 BH
Sample Ref: EXP 122
Job No.905

After fracture surface examination (to be reported in a separate memo) the four samples of sandstones from well 110/12a-1 (Plugs 63,75,95 and 124) were prepared for BS examination. The aim of this work was to investigate the nature of the illite grain coatings and to determine the relationships between the authigenic minerals.

The clay minerals and cements discovered were recognised by their morphology and elemental compositions gained by qualitative microprobe analysis.

Sample preparation

A sample piece of each sandstone was vacuum impregnated with Struers Epofix Resin and subsequently ground and polished. The samples were carbon coated prior to examination in the SEM.

Observations

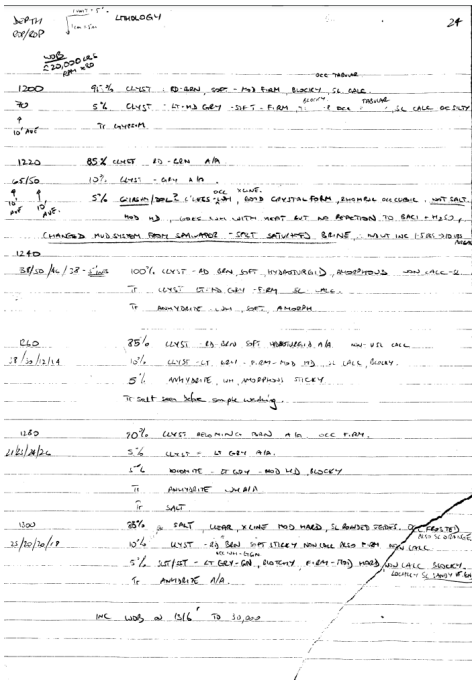
Micrographs taken of the samples are shown in Figures 1 -12.

All four samples were similar and consist of rounded detrital grains with large pore spaces between them. The majority of grains have authigenic illite rim cements growing perpendicular to the margins (Figures 5,6,11,18). In some areas the illite is not perpendicular to the grains (eg Figure 10), however it is not grain tangential so it is still probably authigenic. Occasionally the illite is missing altogether (Figure 2). It is difficult to tell from cross-sectional views whether the clay has a platy or fibrous morphology however a predominantly platy habit is suspected.

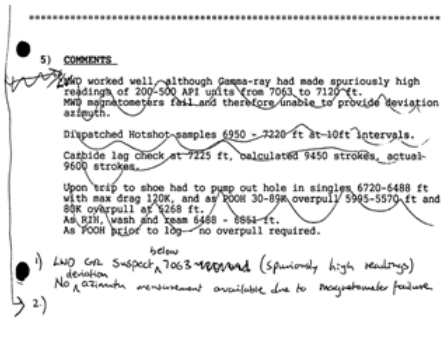
Various euhedral carbonate crystals are found post-dating the illite formation. These include ferroan dolomite, dolomite and ferroan calcite (Figures 5,8,12). Euhedral siderite is also observed replacing occasional grains.

Quartz overgrowths are sometimes seen; again these post-date the illite (Figures 7,9,11). In plug 63 (Figure 1) a ferroan dolomite is engulfed by a quartz overgrowth suggesting that the quartz post-dates the carbonates.

(a) *Page including four paragraphs*



(b) *Page including four paragraphs*



5) COMMENTS

LWD worked well, although Gamma-ray had made spuriously high readings of 200-500 API units from 7063 to 7120 ft. MWD magnetometers fail and therefore unable to provide deviation azimuth.

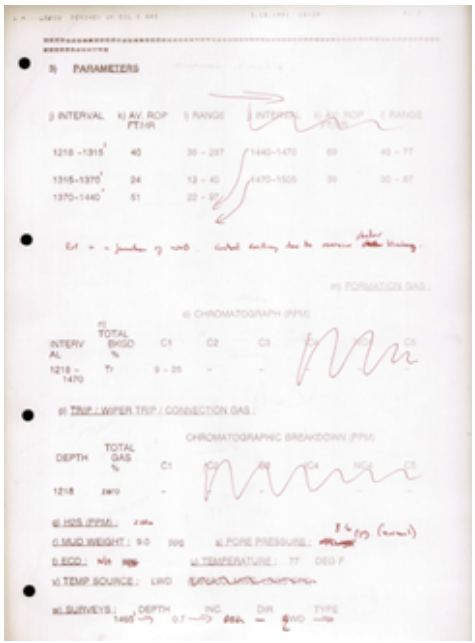
Dispatched Hotshot samples 6950 - 7220 ft at 10ft intervals.

Carbide lag check at 7225 ft, calculated 9450 strokes, actual 9600 strokes.

Upon trip to shoe had to pump out hole in singles 6720-6488 ft with max drag 120K, and at POOH 30-80K overpull 5905-5570 ft and 80K overpull at 5268 ft.
As RIH, wash and ream 6488 - 6851 ft.
As POOH prior to log - no overpull required.

below
1) LWD GR Suspect 7063 ~~ground~~ (spuriously high readings)
deviation
No azimuth measurement available due to magnetometer failure
2)

(c) *Page including seven paragraphs*



(d) *Page including three paragraphs and one table*

Figure 3.4 – *Samples of pages with strike-through text*

documents, or *Non-Dense*, which means this line from the stored structured data is part of a table, figure, cover page, author name, list of contents, etc. These *Dense* and *Non-Dense* labels help us quickly to find the part of the documents which is fundamental to creating a domain-specific corpus without considering the images and tables from the textual context of documents. For example, the text extracted from tables can be ignored to create a corpus since it is often numerical data with no consistent sentence structure.

Dense and *Non-Dense* are coarse-grained layout labels of documents. We could create a more fine-grained layout-level annotation system to identify whether each line is part of a table, figure, footnote, header, or cover page. However, our goal is to create a text corpus from scanned reports. In this context, *Dense* and *Non-Dense* labels will suffice. Due to the heterogeneous structure of these reports, we are facing these main challenges:

— How to explore these documents in a machine-understanding way, which can provide a fine-grained element-level document structure?

— How to automate the process of fine-grained element-level scheme annotation? Generating human-labeled fine-grained element-level scheme annotation for documents and layout labels requires a lot of time and labor.

— How to do layout analysis and assign the layout for each fine-grained element-level scheme stored structure of the documents?

To address these challenges, we propose OCRANA (Optical Character Recognition ANAlytics), a framework to handle documents provided as images or PDF files processed by different OCR engines to build a unified data model. Generally, OCR engine tools process these documents and translate them into an intermediate data format which provides the bounding content boxes. OCRANA processes the intermediate data format and gives a unified and more enriched view of the logical and physical layout using the defined data model. This data model is a configurable multi-level element-based scheme annotation to store the documents. By configurable, we mean data model elements' extendibility (we will discuss it more in Section 3.4.1). This data model is based on three kinds of properties:

1. **Engine-specific** properties extracted directly from the intermediate format as the output of the OCR engine, such as characters, bounding boxes of each character, and page number; These properties are the core properties of the data model.
2. **Statistical** properties are calculated or identified based on the values of engine-specific properties, such as words in each line, bounding boxes of each line, number of words in each line, and the maximum or minimum number of spaces that exist between words in each line. These properties can help define the documents' multi-level scheme structure based on the core properties (Engine-specific properties).

3. **Engine-independent** properties enrich the data model with more text properties, such as POS tags, lemma, and layout labels. These properties are an extendable part of the data model. They can result from statistical, linguistic, or machine-learning models.

This conceptual representation provides a basis for understanding the documents and enables us to have a fine-grained element-level annotation. This representation allows us to assign the layout labels to the line-level structure of documents to provide the basis for further NLP applications.

For automatic recognition of the document structure, it is vital to leverage a kind of supervision to obtain these binary layout labels ("Dense" or "Non-Dense") with minimum effort, to prepare the data for any NLP applications. Due to these reasons, experts have been using weaker types of supervision more frequently, including heuristically creating training data using external knowledge bases, patterns/rules, or other classifiers [46]. Due to the essence of our input (lines of documents), the statistical properties play an important role in distinguishing the *Dense* from *Non-Dense* lines. These properties include the number of words in each line and the minimum and maximum number of spaces between words in each line.

Also, we define a position-based Naïve Bayes algorithm to leverage the semi-supervised methods to have a model for layout labeling. For this semi-supervised method, we need to define some layout labeling functions. Some layout labeling functions are determined based on syntactic information, while others are based on layout information extracted from bounding boxes for each line. These layout labeling functions aim to assign a layout label to each line. Still, with this semi-supervision, we can write a layout labeling function that gives a label to several lines. We use a position-based Naïve Bayes algorithm to generate the labels for groups of lines without any assigned layout labeling. This position-based Naïve Bayes algorithm uses the probability of the positions of each line and the words and symbols of each group of lines related to each layout label to calculate the probability of the new layout labels given the new lines. This way, we assign a layout label to each line of our data store and easily filter the "Dense" lines to generate a domain-specific corpus. As it is clear, the training data is labeled here automatically without human-labor effort by leveraging the weak supervision method.

Our approach is scalable and flexible for significant data processing since we implement the data model in a Parquet [28] columnar datastore in Spark as a distributed environment. Also, the current implementation of OCRANA handles JSON (JavaScript Object Notation) representation of OCR documents. Thanks to different tools, this format can be provided as input, but we chose to rely on Google Vision API [1] [47].

The main contributions of our research are:

1. OCRANA is a scalable framework that efficiently transforms heterogeneous

1. https://cloud.google.com/vision

PDFs or image documents processed by different OCR engines into unified structured information to prepare them for further analysis.

2. OCRANA relies on a unified data model that allows the representation of different kinds of structures of texts and their visual and content-based properties in a fine-grained element-level scheme.
3. OCRANA uses a position-based Naïve Bayes algorithm to construct a large-scale dataset using a weak supervision approach. It enables models to integrate textual and layout information to recognize the different parts of a document. Our results show an improvement in comparison with the Naïve Bayes algorithm model.

The chapter is organized as follows: Section 3.2 reviews the related studies on document image understanding and the available tools. Section 3.3 presents a brief overview of OCRANA. Sections 3.4 and 3.5 detail the building blocks of OCRANA. Section 3.5.4 illustrates the results of our experiments. Section 3.6 briefly presents further semantic analysis as an application of the OCRANA model, and finally, Section 3.7 presents the contributions and conclusion and draws some future steps.

3.2 . Related Work

Document image understanding is the process that transforms the informative content of a document from a paper version into an electronic format detailing its logical content. This process is named *zoning* [48]. Initially, the term *zoning* was introduced to define the process that identifies the text regions and their order of reading in a document image. A general document reading framework would involve segmenting text regions with different roles in the document context. For example, a text column differs from a caption, and a caption differs from a paragraph or footnote. In addition, a complete comprehension of a document includes the analysis of non-textual components, such as drawings, pictures, and mathematical equations. These elements must be separated and treated with suitable modules for further analysis. The zoning method was a combination of the study of geometry (physical) and logical layouts. The geometric structure of the document will be defined by the geometric layout analysis. This phase includes several processes: some preprocessing steps and the decomposition of pages. This last step aims to decompose the document image into maximum homogeneous regions whose elements belong to different data types. The logical layout analysis seeks to define the different logical roles of the detected areas and relationships among them (titles, paragraphs, captions, headers, ...) [41][42][43] [44][45][48]. Over time, the algorithms for layout analysis could be mainly categorized into two groups according to their approach. Bottom-up algorithms start with the minor document components (pixels or connected components) and group them repeatedly to form more significant, homogeneous regions. In comparison, top-down algorithms be-

gin with the complete document image and continually divide it to create smaller and smaller regions. Each approach has its advantage and works well in particular circumstances [44]. Therefore, one may also use a hybrid strategy using a mix of top-down and bottom-up approaches [44]. Lots of other work have been done, and different formats have been suggested to reflect the outcome of OCR document processing, which can generally be categorized as follows [37]: 1) *Logical formats* for users to operate directly with OCR final results, such as HTML, LaTeX, and Microsoft Word.; 2) *OCR engine-specific formats* are the final output of specific engines such as XML, JSON, and hOCR; 3) *benchmarking formats* are proposed for different benchmark aspects of OCR systems. Also, some OCR engines can have multiple outputs.

Text detection identifies and classifies an image document's textual elements. Current text detection approaches handled by common OCR engines can be divided into three groups:

1. **Region-based approaches** that use text-based similarity parameters such as color, height, width, edge, and gradient information to collect pixels;
2. **Texture-based approaches** that use the distinct textural criteria of text sections to extract potential sub-windows and then combine these sub-windows to create the final results;
3. **Hybrid approaches** combine the advantages of region-based approaches that can cover text regions closely with texture-based approaches that can estimate coarse text position in images.

Despite intensive research in document layout analysis, the results still need to be closer to the desired objective, which is a general technique for properly and automatically processing images belonging to various document classes. Google Vision API gives an OCR engine-specific format (JSON) and reshapes its output format to handle large amounts of data. This API uses a hybrid approach to detect texts. The Google Vision API can take images of handwritten texts and PDFs to detect and extract text from them. The extracted string, words, and their bounding boxes are shown in Figure 3.5. Information such as page, block, paragraph, word, and breaks are included in the extracted text in the output of vision API in the JSON format [49][50].

OCRANA is a framework that builds structured information from texts of PDF documents and images processed by OCR engines. In the OCRANA framework, any OCR engine which provides the OCR engine-specific formats with the bounding boxes for characters or tokens can be used, such as Google Vision API, Tesseract, or OCRopus. The purpose of OCRANA is to provide unified and richer structured data of different granularity, such as lines, words, and dense parts of documents, as a basis for advanced analytics of document intelligent understanding.

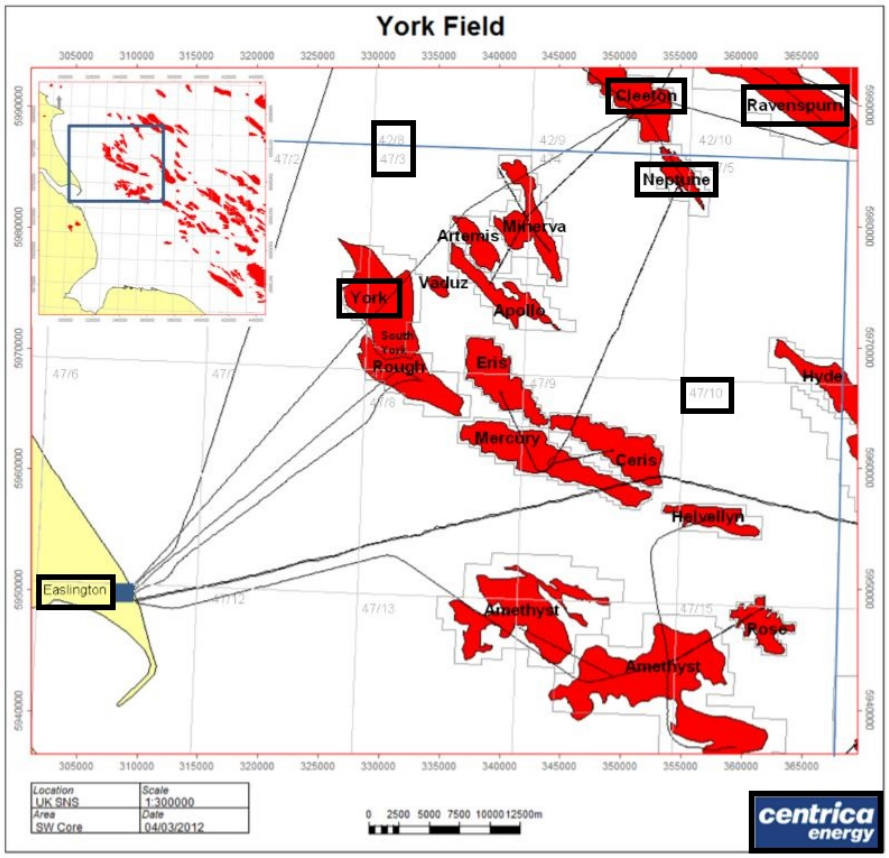


Figure 3.5 – *Example of bounding boxes inside an image*

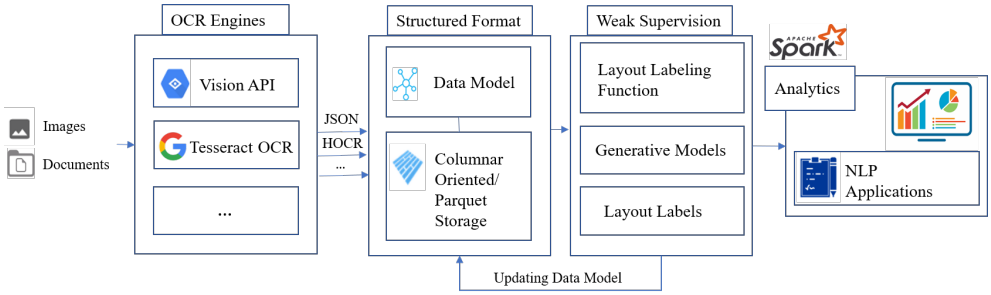


Figure 3.6 – *OCRANA Framework*

3.3 . OCRANA Overview

OCRANA workflow can be summarized in three main steps, as shown in Figure 3.6: 1) OCR engine processing, 2) Conceptual data modeling and storage, and 3) Weak data supervision. The whole OCRANA pipeline is explained as follows.

1. Heterogeneous PDF documents and images are given to the system as input;
2. An OCR engine is applied to the given input to generate an intermediate semi-structured format (JSON, XML, hOCR, ...) which contains the position of characters in the resources (such as bounding boxes);
3. The intermediate results are translated, enriched, and stored according to the OCRANA data model in a columnar format in a distributed environment;
4. Weak supervision methods are applied, and the final layout label is assigned to different data model elements;
5. By taking advantage of the given structured model, NLP analytics can be done.

Figure 3.7 shows an example of the result of each component of the OCRANA workflow. The input PDF document generates the intermediate JSON format using an OCR engine: Google Vision API. The bounding boxes of each OCRANA data model are then built and stored in the Parquet database. By considering a word from the PDF document in section 1 of Figure 3.7 (e.g., STAG), the JSON format of Google vision API contains the separated characters of the word with their corresponding values (e.g., bounding boxes, language, and confidence rate separately for each character (S, T, A, and G)). Section 2 of Figure 3.7shows that the related data model features (Token and Line) are built as is detailed in section 3.4.1. Parquet database containing the tokens and lines based on the OCRANA data model allows storing the structured information.

The OCRANA approach is generic and modular. It processes any OCR engine (such as Vision API, Tesseract OCR), can use any intermediate format (JSON, XML, hOCR, ...), and can store the data model in any columnar datastore (Parquet, Cassandra, ...). In the current implementation, we choose Parquet [51] data

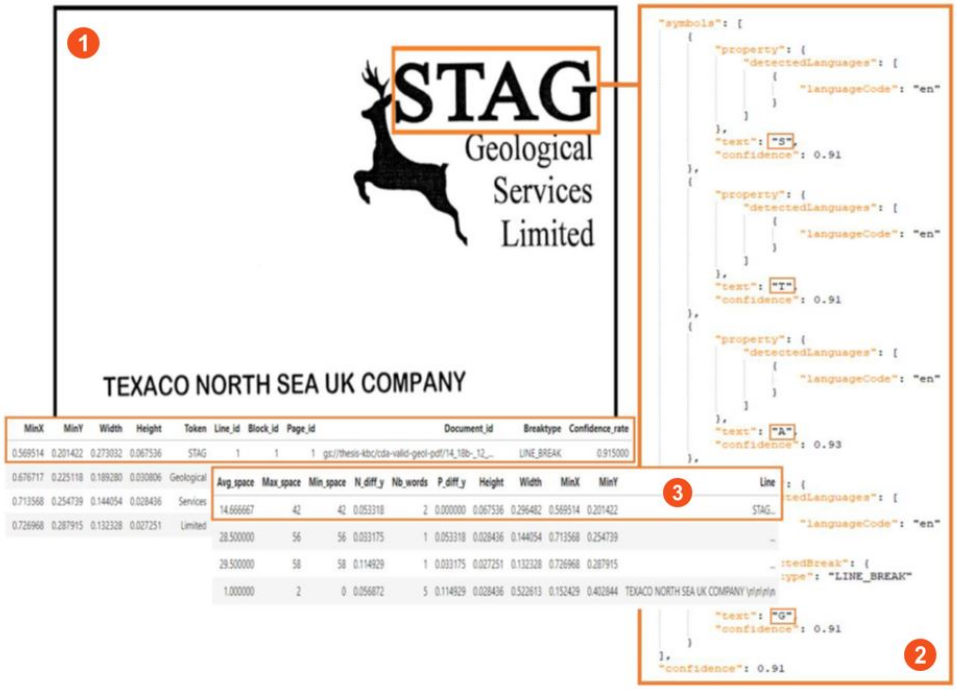


Figure 3.7 – *An example workflow of formats: 1) PDF files and scanned images; 2) Intermediate JSON format; 3) Structured Format*

storage. We rely on an end-to-end process that can use Spark-MLIB or existing python machine learning libraries for the analytics tasks.

3.4 . Structured Format

In this section, we present the OCRANA data model, then explain the implementation of this data model in a parquet dataset to keep the structure of report documents.

3.4.1 . OCRANA Data Model

The OCRANA data model shown in Figure 3.8 comprises a set of elements organized according to a hierarchy that allows a straightforward interpretation of the documents under analysis. The data model reflects but also enriches the tree structure provided by the OCR engine. For example, the data model contains the textual contents and their visual, statistical, and text properties which can guide us in better analysis. This data model is flexible and can handle N-to-N cardinalities between its elements.

The *Document*, which includes a set of *Page*s, is the foundation of the data model. Each *Page* includes a set of *Block*s. A *Block* breaks down into *Paragraph*s, and each *Paragraph* is parsed into *Line*s. Each *Line* has its *Token*s, an aggregation

of characters. The output of the OCR engine captures the position of each page, block, paragraph, and character in the form of a bounding box. Finally, in our data model, a *Token* is the element of a data model with the finest granularity.

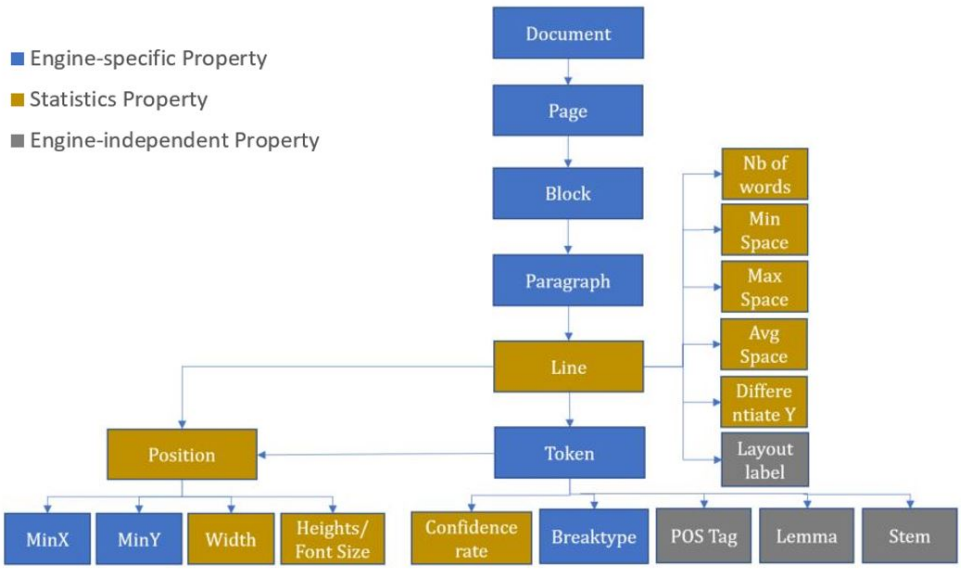


Figure 3.8 – *OCRANA Data Model*

More precisely, a token is any part of the text surrounded by *Space*s (*Breaktype*s that identify a space). Each *Token* is defined by its bounding box position (*Minx*, *MinY*, *Width*, and *Height*), the *BreakType*, and *Confidence rate*.

Minx and *MinY* are initial bounding boxes vertex of the first character of a distinguished token. The *Width*, and *Height* (that can also be considered as the font size) for each token were calculated based on the vertices of the last character in a token.

The *confidence rate* is a value in the output of OCR engines and, in most cases, is provided for characters. It measures an OCR engine's confidence in recognizing characters from an image. OCRANA considers the average of the confidence rates of all the characters of each token as the confidence rate of that token. The *confidence rate* can identify the tokens with a low confidence rate which then contain some characters possibly not correctly distinguished by the OCR engine (low quality of the image documents, ...). This information can help preprocess and clean the extracted texts to have higher-quality texts. The mentioned features (in blue color) are **engine-specific properties** extracted directly from the intermediate format.

In the OCRANA data model a *Line* is characterized by its position (*MinX* and *MinY*) and by *DifferentiateY* that remarks the first token having a *MinY* considerable different with respect to the previous tokens. Moreover, each *Line* has a number of words (*Nb of words*), a minimum number of spaces between two to-

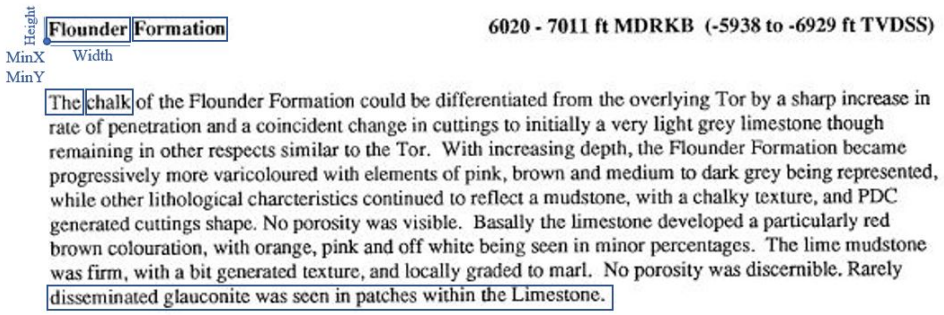


Figure 3.9 – *Captured Position for Tokens and Lines*

kens (*Min Space*), a maximum number of spaces between two tokens (*Max Space*), and the average number of spaces between two tokens (*Avg Space*). When the lines are identified, OCRANA considers the *position* and *height* of each *token* on each *page*. Identifying the lines based on *DifferentiateY* may seem simple, but it can efficiently detect straight lines of a document. Note that OCRANA does not support curved line detection. Though, detecting curved text without a performance degradation on linear text detection is an area of interest for researchers [47]. Figure 3.9 shows an example of a possible bounding box for a line. Due to the need for some calculations to achieve these properties, we call them **statistics properties**, and they are shown in yellow in the OCRANA data model.

OCRANA data model is also extended using **engine-independent properties** mainly related to text processing, such as *POS tag*s, *lemma*s, and *stem*s. According to the granularity, these properties could be for a token (e.g., *POS tag*s) or a line (e.g., *Layout Label*s). Figure 3.8 shows these engine-independent features in gray color. The final *Layout Label*s are assigned based on the heuristic method by using **engine-specific properties** and **statistics properties** to a line; therefore, to its tokens, it is an **engine-independent property**. Any extra information which we are interested in adding to our data model and extending its elements based on our NLP analysis is in this category.

3.4.2 . Column-oriented/Parquet Datastore

Physically in our platform, the data model has been implemented in Parquet columnar datastore. We store hundreds to thousands of documents, each containing thousands to millions of tokens (or even more). We need to record in OCRANA data model tokens and their properties. Processing such datasets efficiently usually require large-scale parallelization to ensure good performances. Parquet is also considered a massively parallel processing system progressively prone to splitting into independent jobs if the query calculation is associated with a single column at once. Our data can be divided in a customized way. We can finally recall one

or more split chunks for further analytics queries, and the usage of Parquet in this context enables us to handle big data [28] [52].
Figures 3.10 and 3.11 show the structured format of a page in Parquet data storage by parsing the intermediate format through the OCRANA data model. Any data model elements are stored as a column in Parquet data format. The token granularity and line granularity reflect the OCRANA data model.

OCRANA is a framework for preparing PDF documents and images for applying advanced analytics. The analytics application includes machine learning methods, natural language processing (NLP), and statistics. There are many tools and ready libraries on NLP and machine learning (e.g., NLTK, SpaCy, Gensim, SparkNLP, Scikit-learn, ...), which can be applied directly to the OCRANA's datastore. We show analytics can be done effectively and efficiently by using existing libraries. For example, the part of speech (POS) tags and the extended features of each token (e.g., lemma, syntactic dependency, shape, detailed POS tags, alpha character recognition, stop word recognition, ...) are identified and added as extended OCRANA data model features to the schema. Figure 3.12 shows the selected features of the OCRANA data model.

This kind of analytics which can be added as extended elements to the OCRANA data model, can help us to label each line to find the *layout label*s of document structure. Before going into the details is better first to illustrate why we need training data and how to label them.

3.5 . Layout Labeling

To do further analysis on parsed documents represented according to the OCRANA data model elements, We must be able to distinguish text and its layout from images and scans, which makes it easier to convert paper documents into a digital format and then categorize them. The layout of the text, as we mentioned, can be varied by the granularity, which we define: Fine-grained layout labels such as header, caption, figure, table, paragraph, and coarse-grained layout labels such as *Dense* and *Non-Dense*; *Dense*, which means this line is part of paragraphs of documents or *Non-Dense* which means this line from the stored structured data is part of a table, figure, cover page, author name, list of contents, etc.

This section presents a weak supervision approach we defined to assign coarse-grained *Layout label*s to each line described in the OCRANA data model. . The input for these functions is data extracted from reports stored in the format of the OCRANA data model, including engine-specific, statistic, and engine-independent properties (e.g., the number of words in each line, font size, average spaces, or POS tags (as described in Section 3.4.1)). These assigned labels are weak based on weak supervision methods. When we have our weak-labeled dataset, we train a machine learning model to have our model for document structure recognition. Figure 3.13) shows the whole pipeline.

a

MinX	MinY	Width	Height	Token	Line_id	Block_id	Page_id	Document_id	Breaktype	Confidence_rate
0.147404	0.459716	0.023451	0.014218	The	15	11	11	gs://thesis-kbc/cda-valid-geol-pdf/14_18b-_12_...	SPACE	0.993333
0.174204	0.459716	0.035176	0.015403	chalk	15	11	11	gs://thesis-kbc/cda-valid-geol-pdf/14_18b-_12_...	SPACE	0.992000
0.216080	0.459716	0.015075	0.015403	of	15	11	11	gs://thesis-kbc/cda-valid-geol-pdf/14_18b-_12_...	SPACE	0.990000
0.234506	0.458531	0.020101	0.014218	the	15	11	11	gs://thesis-kbc/cda-valid-geol-pdf/14_18b-_12_...	SPACE	0.990000
0.259631	0.458531	0.060302	0.014218	Flounder	15	11	11	gs://thesis-kbc/cda-valid-geol-pdf/14_18b-_12_...	SPACE	0.993750
0.324958	0.458531	0.070352	0.014218	Formation	15	11	11	gs://thesis-kbc/cda-valid-geol-pdf/14_18b-_12_...	SPACE	0.995556
0.400335	0.458531	0.035176	0.014218	could	15	11	11	gs://thesis-kbc/cda-valid-geol-pdf/14_18b-_12_...	SPACE	0.996000
0.442211	0.458531	0.015075	0.014218	be	15	11	11	gs://thesis-kbc/cda-valid-geol-pdf/14_18b-_12_...	SPACE	1.000000
0.463987	0.458531	0.087102	0.014218	differentiated	15	11	11	gs://thesis-kbc/cda-valid-geol-pdf/14_18b-_12_...	SPACE	0.997143

Figure 3.10 – *Structured format of a document presented based on tokens*

Avg_space	Max_space	Min_space	N_diff_y	Nb_words	P_diff_y	Height	Width	MinX	MinY	Line
2.461538	21	0	0.043839	12	0.041469	0.013033	0.726968	0.144054	0.413507	Flounder Formation 6020 - ...
0.947368	1	0	0.014218	18	0.043839	0.015403	0.718593	0.147404	0.457346	The chalk of the Flounder Formation could be d...
0.944444	1	0	0.014218	17	0.014218	0.013033	0.683417	0.144054	0.471564	rate of penetration and a coincident change in...
0.937500	1	0	0.013033	15	0.014218	0.013033	0.703518	0.145729	0.485782	remaining in other respects similar to the Tor...
0.937500	1	0	0.015403	15	0.013033	0.013033	0.730318	0.144054	0.498815	progressively more varicoloured with elements ...
0.937500	1	0	0.013033	15	0.015403	0.013033	0.693467	0.145729	0.514218	while other lithological charcteristics contin...
0.933333	1	0	0.014218	14	0.013033	0.013033	0.686767	0.147404	0.527251	generated cuttings shape. No porosity was visi...
0.941176	1	0	0.014218	16	0.014218	0.013033	0.720268	0.144054	0.541469	brown colouration, with orange, pink and off w...
0.944444	1	0	0.014218	17	0.014218	0.014218	0.700168	0.142379	0.555687	was firm, with a bit generated texture, and lo...
0.900000	1	0	0.045024	9	0.014218	0.011848	0.457286	0.145729	0.569905	disseminated glauconite was seen in patches wi...

Figure 3.11 – *Structured format of a document presented based on lines*

a. *P_diff_y* and *N_diff_y* in Figure 3.11 represent the *Differentiate Y* in the OCRANA' data model for each line regarding its previous and next line respectively.

Token	Line_id	Page_id	Document_id	Pos_tags	Lemma	Detail_tag	Dependency	Word_shape	Alpha_char	Stop_word
The	15	11	gs://thesis-kbc/cda-valid-geol-pdf/14_18b-_12_...	DET	the	DT	ROOT	Xxx	True	True
chalk	15	11	gs://thesis-kbc/cda-valid-geol-pdf/14_18b-_12_...	PROPN	chalk	NNP	ROOT	xxxx	True	False
of	15	11	gs://thesis-kbc/cda-valid-geol-pdf/14_18b-_12_...	ADP	of	IN	ROOT	xx	True	True
the	15	11	gs://thesis-kbc/cda-valid-geol-pdf/14_18b-_12_...	DET	the	DT	ROOT	xxx	True	True
Flounder	15	11	gs://thesis-kbc/cda-valid-geol-pdf/14_18b-_12_...	PROPN	Flounder	NNP	ROOT	Xxxxx	True	False
Formation	15	11	gs://thesis-kbc/cda-valid-geol-pdf/14_18b-_12_...	NOUN	formation	NN	ROOT	Xxxxx	True	False
could	15	11	gs://thesis-kbc/cda-valid-geol-pdf/14_18b-_12_...	AUX	could	MD	ROOT	xxxx	True	True
be	15	11	gs://thesis-kbc/cda-valid-geol-pdf/14_18b-_12_...	AUX	be	VB	ROOT	xx	True	True
differentiated	15	11	gs://thesis-kbc/cda-valid-geol-pdf/14_18b-_12_...	VERB	differentiate	VBN	ROOT	xxxx	True	False

Figure 3.12 – *NLP analytics related to each token*

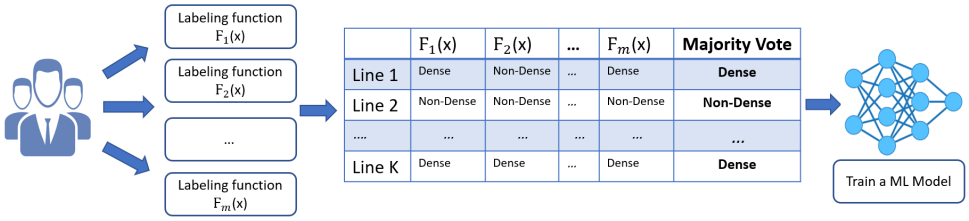


Figure 3.13 – *Weak supervision "pipeline"*

To assign these weak layout labels and the final model, we need to apply these steps:

1. Define the **labeling functions** which can assign a label to each line with high confidence from human opinions. These labeling functions are simple rules that can guess the correct layout label for a line. These labeling functions can use information from the OCRANA elements, such as statistics and engine-independent properties. Also, these OCRANA elements are categorized into two types: (1) The statistic elements such as *Nb of Words*, *Min Space*, *Avg Space*, and *Font Size*, and (2) The extended features of tokens in each line as semantic elements such as *POS Tag* and *Lemma*. As an example for the first category, we can use the minimum number of spaces (*Min Space*) in a line as a rule for our labeling function and say: "If there are more than two gaps (spaces) in a line, the layout label is *Non-Dense* if not, the layout label is *Dense*." We should remember that we can use any number of labeling functions with any information from the data model as far as a human can logically justify it. It concludes that here these labeling functions are domain-independent functions.
2. Some lines can have more than one labeling functions that assign different layout labels to each line. Different methods can produce the best label for each input (line). We could take the majority vote on labels to assign a

final layout label. So, if 3 labeling functions voted for *Non-Dense*, we would assume the label is *Non-Dense*. The problem is when some lines have an even number of layout labels of *Non-Dense* and *Dense* as the output of the assigned labeling functions. In this case, assigning a certain layout label to that line is difficult. Also, some of the rules will have more weight compared with other rules. For example, the rule which says: "If the number of the words is less than 5, then the layout label is *Non-Dense* if not is *Dense*", has more weight than a rule like this: "If there are more than two gaps (spaces) in a line, the layout label is *Non-Dense* if not the layout label is *Dense*." for the lines specifically at the end of a paragraph. There can be two methods to apply these labeling functions:

— Using **majority votes** for final labels of each labeling function; we also can assign the weights to prioritize different labeling functions by human experts [53][54].

— Using a machine learning model such as the Naïve Bayes model, SGD with Gibbs sampling to assign weights to each labeling function. This method is part of the Data programming method, which was first presented in the Snorkel package by Alex Ratner and Chris Re [35] [46][55].

We use the former method: Majority voting. Based on our labeling functions, some lines remain without assigned layout labels. By default, for these lines without any labeling functions, a specific label, "IDK" ("IDK" stands for "I Don't Know"), is assigned.

3. A labeled dataset - weak labels - is generated and can be used exactly as any other labeled dataset by **training a machine learning model** to predict the most relatable layout labels for each line. Here, we define the **position-based Naïve Bayes classifier** as our ML model, which can outperform other models (see Table 3.10).

3.5.1 . Labeling Functions

We defined a set of rules-based functions $f_i(x)$ to assign to OCRANA extracted lines Dense or Non-Dense labels. These rules include the grammar syntax of sentences, if any exists in the lines, or any other statistical information of each line, such as the number of spaces or words inside the lines. Some lines could have more than one labeling function, which leads to more than one label. As we explained in the main steps of assigning weak layout labels to each line in Section 3.5, we use weighted majority votes to give the final label to each line. The human assigns the weights for this majority voting when writing down the labeling functions. Based on the different weighting by experts, we will probably have various layout labels. However, in any case, we face weak labels for each line. We consider the constrained-based labeling functions to assign a more accurate layout label, which logically can structure a layout. Let's look at one example of writing constraint-

based semantic labeling functions. This is an example of the second category of labeling functions using the semantic elements of OCRANA. We will begin by the search for chunks corresponding to individual noun phrases and verb compounds. We believe in the Non-dense parts of texts; in each line, we do not have complete sentences. They are mostly noun phrases. Then, by this hypothesis, we write some regular expressions to find the noun and verbal phases based on POS tag patterns (Detail_tags in Figure 3.12). We already have POS tags as part of engine-specific attributes in our dataset.

```
grammar = r"""
      NP: {<DT>*<NN.*>+<JJ>+<NN.*>*<NN.*>}
          {<DT>*<NN.*>+<NN.*>*<NN.*>}
          {<PRP\$>+<JJ>+<NN?>}
          {<PRP\$>+<NN?>}
          {<JJ>+<NNS>}
          {<DT|PP\$>?<JJ>*<NN>*<DT?>}
          {<NNP>+<DT?>*<_SP?>}
          {<RB?>*<JJ.?>*<.?>*<JJ,?>*<.?>*<NN.*>}
          {<LS?>*<.?>*<NN?>}
      VB: {<VB.*>+<VB.*>}
          {<VB.*>+<NN|NNP?>+<VB.*>}
          {<VB.*>}

    """
```

Figure 3.14 – *The default grammar used as a rule-based chunker*

Figure 3.14shows a predefined grammar as the default grammar for rule-based chunkers. This grammar also can be extended. The focus of the rule-based chunker here is mainly to find the noun and verbal phrase chunks. When we apply this grammar to a sentence based on POS tags of tokens of each sentence, we can use the defined chunks as the subtrees for our sentence.

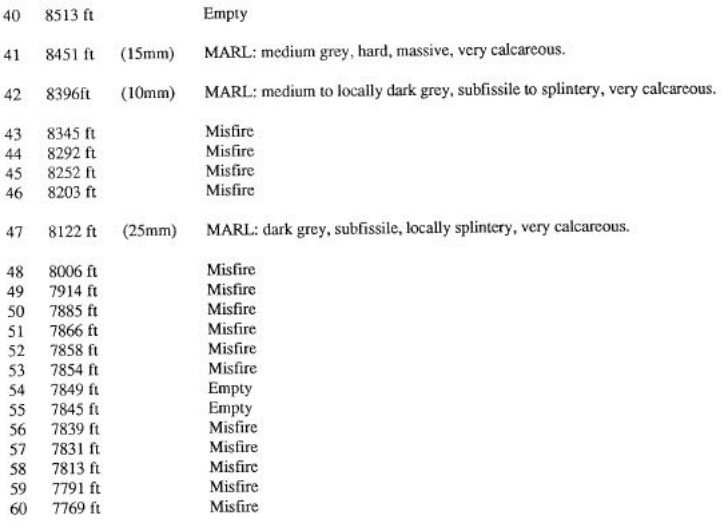
One of the labeling functions here can be if any verbal phrase chunk (VB) exists in one line; it is more likely that the line is part of a dense part of the text.

The final *layout label*s for an example page in Figure 3.15 based on our defined labeling functions are shown in Figure 3.16. We aim to use weak supervision to apply these *layout label*s to all the *line*s in the documents.

Example 6. *Given the page shown in Figure 3.17, a part of the page's information stored in the structured format under the OCRANA model scheme is shown in Table 3.1. Then the input for the Naïve Bayes classifier is the words of each line (Line column in Table 3.1).*

3.5.2 . Weakly Supervised Learning Model

Once the final layout labels are assigned using labeling functions to the lines of documents based on OCRANA elements, we finally have a database with weak labels. Now, it is time to train an ML model to recognize the layout structure of the documents.



40	8513 ft		Empty
41	8451 ft	(15mm)	MARL: medium grey, hard, massive, very calcareous.
42	8396ft	(10mm)	MARL: medium to locally dark grey, subfissile to splintery, very calcareous.
43	8345 ft		Misfire
44	8292 ft		Misfire
45	8252 ft		Misfire
46	8203 ft		Misfire
47	8122 ft	(25mm)	MARL: dark grey, subfissile, locally splintery, very calcareous.
48	8006 ft		Misfire
49	7914 ft		Misfire
50	7885 ft		Misfire
51	7866 ft		Misfire
52	7858 ft		Misfire
53	7854 ft		Misfire
54	7849 ft		Empty
55	7845 ft		Empty
56	7839 ft		Misfire
57	7831 ft		Misfire
58	7813 ft		Misfire
59	7791 ft		Misfire
60	7769 ft		Misfire

Figure 3.15 – *Non-dense part of a text in a PDF document*

Avg_space	Max_space	Min_space	N_diff_y	Nb_words	P_diff_y	Height	Width	MinX	MinY	Line	Layout_Label
6.750000	14	14	0.013033	3	0.000000	0.011848	0.211055	0.180905	0.125592	8625 ft Misfire	Non_Dense
6.750000	14	14	0.014218	3	0.013033	0.011848	0.211055	0.180905	0.138626	8609 ft Misfire	Non_Dense
6.750000	14	14	0.015403	3	0.014218	0.011848	0.212730	0.180905	0.152844	8588 ft Misfire	Non_Dense
6.750000	14	14	0.013033	3	0.015403	0.010664	0.214405	0.179229	0.168246	8513 ft Misfire	Non_Dense
6.750000	14	14	0.014218	3	0.013033	0.011848	0.212730	0.179229	0.181280	8345 ft Misfire	Non_Dense
6.750000	14	14	0.013033	3	0.014218	0.010664	0.194305	0.180905	0.195498	8292 ft Lost	Non_Dense
5.200000	11	10	0.028436	4	0.013033	0.011848	0.254606	0.140704	0.208531	18 8252 ft Misfire \n	Non_Dense
1.866667	10	10	0.016588	14	0.028436	0.014218	0.723618	0.142379	0.236967	19 8203ft (15mm) MARL: light to ...	Non_Dense
0.750000	1	0	0.027251	3	0.016588	0.011848	0.418760	0.038526	0.253555	- very calcareous. \n	Non_Dense
6.750000	14	14	0.015403	3	0.027251	0.010664	0.214405	0.182580	0.280806	8006 ft Misfire	Non_Dense
6.750000	14	14	0.009479	3	0.015403	0.013033	0.211055	0.184255	0.296209	7914 ft Misfire	Non_Dense
5.400000	14	0	0.016588	4	0.009479	0.010664	0.358459	0.038526	0.305687	- 7885 ft Misfire	Non_Dense
6.750000	14	14	0.026066	3	0.016588	0.011848	0.211055	0.185930	0.322275	7866 ft Misfire \n	Non_Dense
2.333333	14	14	0.005924	11	0.026066	0.014218	0.680067	0.185930	0.348341	7858ft (15mm) LIMESTONE: ligh...	Non_Dense

Figure 3.16 – *Layout labels for Non-dense parts of the text*

Naïve Bayes classifier

Using machine learning, a classifier is a model that uses particular attributes to distinguish between various objects. Using Bayes theorem, It is possible to calculate the likelihood of Y occurring given the occurrence of X. The probabilistic machine learning models known as Naïve Bayes classifiers are used for classification tasks. In other words, Bayes' theorem describes updating the probabilities of hypotheses (guesses) when given evidence. A probabilistic classifier's output P(Y|X) is the probability that an input X belongs to a class Y. The input is a set of lines. Each line belongs to a different layout label: *Dense* or *Non-Dense*. Each line contains tokens that are given probabilities based on the number of occurrences within that line. For example, a word with two or three numerical values in a line belongs to

Algorithm 1 The Naïve Bayes classifier algorithm

Input: C (classes for each line), L set of lines, V vocabulary of L
Output: loglikelihood , logprior , V each class c $\in$ C ▷ Calculate P(c) terms
N_{word} = number of lines in L
N_c = number of lines from L in class c
$logprior[c] \leftarrow \frac{\log N_c}{\log N_{word}}$
$V \leftarrow$ vocabulary of L
$newdoc[c] \leftarrow$ append(l) l $\in$ L with class c each word w $\in$ V ▷ Calculate P(w | c) terms
$count(w, c) \leftarrow$ # of occurrences of w in $newdoc[c]$
$loglikelihood[w, c] \leftarrow \log \frac{count(w,c)+1}{\sum_{w \in V}(count(w,c)+1)}$
return $loglikelihood, logprior, V$

Figure 3.17 – *Part of a page with different lengths of lines*

Table 3.1 – *Part of the stored structured format of a page*

lineindex	*Height*	*Width*	*MinX*	*MinY*	Line
0	0.016647	0.097479	0.463866	0.1200095	TALISMAN
1	0.016647	0.087329	0.463866	0.1305075	SINOPEC
2	0.013080	0.070588	0.463866	0.145065	ENERGY UK
3	0.040428	0.569748	0.216807	0.244947	Wireline Logging QA/QC Report
4	0.017836	0.021849	0.487395	0.278240	on
5	0.030916	0.205042	0.400000	0.325803	Baker Hughes

a *Non-Dense* class with a higher probability. Naïve Bayes can learn the pattern of analyzing a collection of documents that have been classified and comparing the contents in all classes by constructing a list of words and their frequency. New lines of text are categorized using a list like this according to the highest posterior probability.

Naïve Bayes is a generative model because Eq. 3.1 states a sort of implicit hypothesis about the generation of a document: first, a class is sampled from P(x), and then the words are formed by sampling from P(x|y). By following this process, we can create fake documents, or at the very least, their word counts [56].

$$\hat{y} = \underset{y \in Y}{\operatorname{argmax}} P(y|x) = \underset{y \in Y}{\operatorname{argmax}} P(x|y)P(x) = \operatorname{argmax} likelihood \times prior \quad (3.1)$$

Naïve Bayes classifier makes two simple assumptions: (1) **bag of words assumption** , which means we assume that the order of appearing words does not matter (e.g., the word "formation" has the same effect on the classification result whether it occurs as the 1st, middle, or last word in the input for classification). Therefore, we assume that the features only encode word identity and not the position of the words; (2) **Naïve Bayes assumption** emphasizes the conditional independence assumption that the probabilities $P(x_i|Y)$ are independent given the class Y. The chain rule can be used to expand the Naïve Bayes assumption as follows:

$$P(x_1, x_2, ..., x_n|Y) = P(x_1|y)P(x_2|y)...P(x_n|y) \quad (3.2)$$

Word positions must be taken into consideration when using the Nave Bayes classifier on text:

$$y = \underset{y \in Y}{\operatorname{argmax}} P(y) \prod_{i \in \text{word orders}} P(w_i|y) \quad (3.3)$$

Calculations for Naïve Bayes and language modeling are carried out in log space to prevent underflow and increase processing speed. Therefore, Eq. 3.3 is typically represented as:

Word orders $\leftarrow$ all word positions in the test document

$$y = \underset{y \in Y}{\operatorname{argmax}} \log P(y) + \sum_{i \in \text{word orders}} \log P(w_i|y) \quad (3.4)$$

Eq. 3.4 computes the predicted class as a linear function of input features by considering features in log space. The Naïve Bayes classifier pseudo code algorithm can be seen in Algorithm 1.

Table 3.2 – *The new stored structured format of a page after the transformation of the value of the bounding boxes*

lineindex	*Height*	*Width*	*MinX*	*MinY*	Line
0	-4.0	-2.0	-1.0	-2.0	TALISMAN
1	-4.0	-2.0	-1.0	-2.0	SINOPEC
2	-4.0	-3.0	-1.0	-2.0	ENERGY UK
3	-3.0	-1.0	-2.0	-1.0	Wireline Logging QA/QC Report
4	-4.0	-4.0	-1.0	-1.0	on
5	-3.0	-2.0	-1.0	-1.0	Baker Hughes

Table 3.3 – *The new stored structured format of a page after the transformation of the value of the bounding boxes*

lineindex	*Height*	*Width*	*MinX*	*MinY*	Line
0	w_h_-4.0	w_w_-2.0	w_x_-1.0	w_y_-2.0	TALISMAN
1	w_h_-4.0	w_w_-2.0	w_x_-1.0	w_y_-2.0	SINOPEC
2	w_h_-4.0	w_w_-3.0	w_x_-1.0	w_y_-2.0	ENERGY UK
3	w_h_-3.0	w_w_-1.0	w_x_-2.0	w_y_-1.0	Wireline Logging QA/QC Report
4	w_h_-4.0	w_w_-4.0	w_x_-1.0	w_y_-1.0	on
5	w_h_-3.0	w_w_-2.0	w_x_-1.0	w_y_-1.0	Baker Hughes

Table 3.4 – *The new stored structured format of lines for our position based Naïve Bayes classifier*

lineindex	New Line
0	w_h_-4.0 w_w_-2.0 w_x_-1.0 w_y_-2.0 TALISMAN
1	w_h_-4.0 w_w_-2.0 w_x_-1.0 w_y_-2.0 SINOPEC
2	w_h_-4.0 w_w_-3.0 w_x_-1.0 w_y_-2.0 ENERGY UK
3	w_h_-3.0 w_w_-1.0 w_x_-2.0 w_y_-1.0 Wireline Logging QA/QC Report
4	w_h_-4.0 w_w_-4.0 w_x_-1.0 w_y_-1.0 on
5	w_h_-3.0 w_w_-2.0 w_x_-1.0 w_y_-1.0 Baker Hughes

3.5.3 . Position-based Naïve Bayes Classifier

The core of the position-based Naïve Bayes classifier is the Naïve Bayes algorithm. However, in the position-based Naïve Bayes classifier, $loglikelihood, logprior$ would be calculated not just by a set of words of each line related to a specific class but also by the values of bounding boxes related to each line (*Width*, *Height*, *MinX*, *MinY*). This information is stored in OCRANA per each line. To add this numerical value as input of the Naïve Bayes algorithm, first, we need to cast this numerical value to categorical ones. We used the log transform function. Highly skewed distributions can become less skewed by applying the log transformation. However, the most important reason here is to map very close values to the same final value. Finally, this value will be considered a unique string and be added to the vocabulary of each line. The pseudo-code algorithm for Naïve Bayes and position-based Naïve Bayes classifier algorithm is shown in Algorithm 1 and 2, respectively.

Algorithm 2 The position based Naïve Bayes classifier algorithm

Input: C (classes for each line), L set of lines, V vocabulary of L, Bounding boxes of each line (*Width*, *Height*, *MinX*, *MinY*)
Output: loglikelihood, logprior, V each class c $\in$ C ▷ Calculate P(c) terms
N_{word} = number of lines in L
N_c = number of lines from L in class c
$logprior[c] \leftarrow \frac{\log N_c}{\log N_{word}}$
$V \leftarrow$ vocabulary of L
$V \leftarrow$ Round(logtransform(x)) when x $\in$ (*Width*, *Height*, *MinX*, *MinY*)
$newdoc[c] \leftarrow$ append(l) l $\in$ L with class c each word w $\in$ V ▷ Calculate P(w | c) terms
$count(w, c) \leftarrow \#$ of occurrences of w in $newdoc[c]$
$loglikelihood[w, c] \leftarrow \log \frac{count(w,c)+1}{\sum_{w \in V}(count(w,c)+1)}$
return $loglikelihood, logprior, V$

Example 7. *Given the following Width, Height, MinX, and MinY in Table 3.1 for the consecutive lines of Page in Figure 3.17. The transition steps to prepare the input for position based Naïve Bayes classifier algorithm are shown as follows:*

- *First, we need to calculate the Round of log transform of the values of bounding boxes for each line as it is indicated in Algorithm 2 as follows:*

$$V \leftarrow \textit{Round(logtransform(x)) when x} \in (\textit{Width}, \textit{Height}, \textit{MinX}, \textit{MinY}) \tag{3.5}$$

The transformed value is shown in Table 3.2.

Table 3.5 – *Comparison of validation set labels: manual vs. weak labels*

Name of Set	Dense	Non_Dense	IDK	Total
Manually (Golden labels)	774	5149	0	5923
Labeling functions	770	5141	12	5923

- *To not confuse that the new values are supposed to have a categorical essence as the input of our Naïve Bayes algorithm, we add w_h_, w_w_, w_x_, and w_y_ in front of each transformed value coming from Height, Width, MinX, and MinY, respectively. Then it will be much easier for us to know the transformed values defined as brackets of numerical value with the assigned transform value. The new categorical value is shown in Table 3.3.*
- *Now, It's time to create new lines as input for our position-based Naïve Bayes classifier algorithm. The New Line column in Table 3.4is the input for the Naïve Bayes classifier to calculate the loglikelihood and logprior for the words of each line and the assigned category of positions for each line.*

3.5.4 . Evaluation and Experiments

In this section, first, we evaluate the position-based Naïve Bayes classifier to model the layout labels of the document. Then, we show the result of some domain-specific information extracted from records.

Dataset description

This dataset consists of 3459 documents with a total number of 166,548 pages and a total number of 7,037,338 lines with two types of layout labels (1,149,271 dense lines, 5,872,569 Non_Dense lines, and 15,498 lines without labels (IDK)). It can be easily seen that the distribution of layout labels in geological reports is very biased toward Non-dense labels, which is logical because most of the report pages contain seismic diagrams, tables of geophysical values, and figures. Then even if we expect the logprior in the Naïve Bayes model to be more biased toward the Non-dense class, as seen in Table 3.9, this assumption is correct. The validation set is generated by labeling the lines manually. This set is randomly chosen from 100 pages from 100 random documents. Even though the proportion of validation set to train set is very low, it provides a golden label set to compare the different models.

When these lines are labelled manually, we can see the quality of provided weak labels through our labeling functions. Twelve rows initially did not receive any assigned labels through our labeling functions, and as can be seen, our labeling functions provide very accurate and high-quality labels. In Table 3.5, we can see the comparison of two validation sets, one with golden labels provided by labeling manually and another one with labels assigned with labeling functions.

5,923 random samples from the whole 7,037,338 lines is a small portion. Still, it gives us a base to understand the quality of weak layout labels for lines generated

Table 3.6 – *The quality of weak labels on the validation set in comparison with manual labels*

Model	Recall	Precision	F1 Score
Manually labeled validation set	0.98	0.97	0.97

through our labeling functions, which almost have high accuracy and coverage of over 97%, as shown in Table 3.6.

Figures 3.18 and 3.19 show how the pages and lines are distributed on separate files. As can be seen, most documents have fewer than 100 pages and generally contain fewer than 5000 lines.

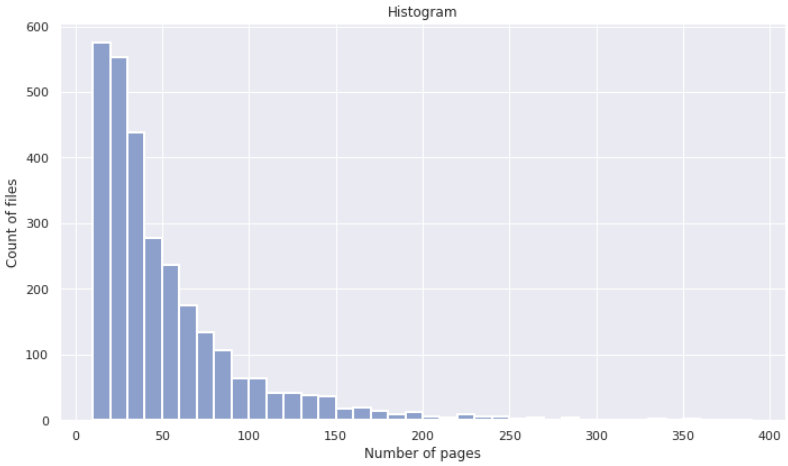


Figure 3.18 – *The dispersion of pages per file*

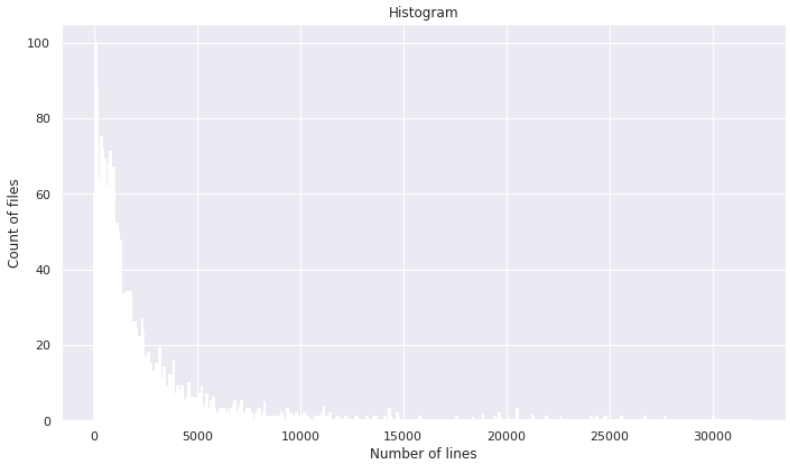


Figure 3.19 – *The dispersion of lines per file*

Now that we have our labeled dataset, it is evident that we face an imbalanced classification problem. However, considering the Naïve Bayes method, this imba-

lanced classification probably should help distinguish a more specific likelihood for words appearing in each class. Then we assume a higher amount of samples and a more significant imbalance should support our method. To see that this imbalanced data affects on the final result of our position-based Naïve Bayes method, we evaluate our position-based Naïve Bayes method on three datasets generated from our labeled dataset. First, we randomly choose about 1,200,000 samples for our training and 300,000 for our test dataset (we call this dataset "*Sample dataset 1*") - we keep the test set the same in all our experiments. Then, we chose a sample of 3,000,000 lines as our second training set ("*Sample dataset 2*"), and in our final try, we chose all records of the labeled dataset ("*whole dataset*"). More details can be seen in Table 3.7.

Table 3.7 – *Information about training sets to seek the impact of the number of samples and the imbalance classes*

datasets	Train		Total number
	Dense	Non_Dense	
Sample dataset 1	400,000	800,000	1,200,000
Sample dataset 2	1,049,271	1,600,000	2,649,271
Whole dataset	1,049,271	5,672,569	6,721,840

We have applied Naïve Bayes and position-based Naïve Bayes methods on these defined datasets and calculated the accuracy for the test dataset to examine our assumption about imbalance classes and the number of samples. The accuracy is presented in Table 3.8. As you can see, the best accuracy belongs to the "*Whole dataset*", which has a higher number of samples and a more significant gap in imbalanced classes. Then we keep the model which is trained on this training set as our default model, and in the next section (Section 3.5.4), when we talk about Naïve Bayes and position-based Naïve Bayes model, we mean the model which is trained on the "*Whole dataset*". Since the "*Whole dataset*" has the more significant class imbalanced towards *Non_Dense* class, we expect a higher logprior (identification of the initial bias and tendency towards a specific class) towards *Non_Dense* class which, as it is shown in Table 3.9, this assumption is correct.

Table 3.8 – *Information about training sets to seek the impact of the number of samples and the imbalance classes*

Method	Accuracy		
	Sample dataset 1	Sample dataset 2	Whole dataset
Naïve Bayes	0.7850	0.8424	0.8651
Position-based Naïve Bayes	0.8644	0.8934	0.9154

Settings

We used python version 3.7.4 for our implementation. As we mentioned, the input of our algorithms is the lines in the parquet format. We may use Spark on

Table 3.9 – *Calculated logprior for three different training sets, 0 is the value for neutral bias, then value under zero shows a tendency to label the lines as Non_Dense*

	Sample dataset 1	Sample dataset 2	Whole dataset
logprior	-0.6931	-0.4219	-1.6875

top of the parquet format. In our context, the data model implemented in parquet format can be directly given as input to the Spark framework for big data analytics. Big data analytics can be applied directly to Parquet since it is a massively parallel processing system to handle big data.

Parquet is a columnar data store that is free, open-source, optimized, and developed on the Apache Hadoop platform [34].

Spark is a prominent open-source distributed processing framework for big data analytics. Spark offers modules for streaming, structured data processing, graph analysis, and machine learning. To facilitate rapid application development in a variety of languages, such as Scala, Java, Python, R, and C++, Spark offers language-integrated APIs [57] [58]. Spark shows fascinating performances at a large scale if implemented on top of a columnar-based data format such as Parquet [51].

Evaluation Metrics

We used classical evaluation metrics for binary classification in a non-balanced dataset, including Precision, Recall or Sensitivity, Specificity and F1-score. Accuracy could only be a good measure if the dataset is balanced. Using accuracy to perform the models in imbalanced dataset scenarios can result in a misleading interpretation of results. Then let's have a look at the evaluation metrics which we used.

Recall sometimes known as sensitivity or true positive rate (TPR), is used as one of the metrics. Recall is the probability that an actual positive will test, which means how well a test detects the positives. This can be maximized by a cheating test that always returns "positive". Recall is defined as follows:

$$Recall/Sensitivity\ (TPR) = \frac{TP}{TP + FN} \tag{3.6}$$

We used another metric called specificity, known as the true negative rate. Specificity is the probability that an actual negative will test negative, which means how good a test is at avoiding false alarms. This can be maximized by a cheating test that always returns "negative". Specificity is defined as follows:

$$Specificity\ (TNR) = \frac{TN}{TN + FP} \tag{3.7}$$

We used another important metric called precision, also known as positive predictive value. Precision is the ratio of correctly positive labels to all positively predicted

Table 3.10 – *The performance of different models on validation set with manual labels*

Model	Recall (TPR)	TNR	Precision	F1 Score
Naïve Bayes classifier	0.8372	0.9594	0.9434	0.8871
Position based Naïve Bayes classifier	0.8449	0.9673	0.9514	0.8949
Bert classifier	0.7974	0.8911	0.8226	0.8086

labels. Precision is defined as follows:

$$Precision = \frac{TP}{TP + FP} \tag{3.8}$$

Last but not least, we used an important metric called the F1 score. It is the harmonic mean of recall and precision. Both false positives and false negatives are considered. Therefore, it performs well when used with an unbalanced dataset. F1 score is defined as follows:

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.9}$$

Position based Naïve Bayes classifier performance

To see how the position-based Naïve Bayes classifier performs well in assigning a layout label to each line, we compare the model performance based on the metrics presented in section 3.5.4. We generated a manually-labeled validation set. This set is randomly generated from 100 pages of 100 randomly chosen documents. More information about this validation set can be seen in both tables 3.5 and 3.6. We see the performance of 3 different models on this validation set: the Naïve Bayes classifier, the position-based Naïve Bayes classifier, and the BERT classifier. We have already explained the first two; the last classifier, which we will explain in detail in the next chapter (Section 4.2.2), is a word representation method. However, as a classification technique, a classification layer may be added on top of the word embedding layers. As can be seen, the performance of the BERT classifier is lower than others. We can think of two reasons: (1) we use the existing public BERT model pre-trained, which is trained in a public domain, and many out-of-vocabulary words (OOV) exist in the subsurface domain. OOVs are the words that do not exist in the pre-train words dictionary (we explain more about OOV in detail in Chapter 4)), (2) the only consideration for the BERT classifier similar to the Naïve Bayes classifier is the words in the lines, not extra information such as their position in the document. Also, as shown in Table 3.10, Naïve Bayes and position-based Naïve Bayes classifiers have much better performance on Specificity (TNR) than the BERT classifier, which means the Naïve Bayes classifier, in general, works better at avoiding false alarms.

Now that we have the final model (Position based Naïve Bayes classifier), we apply the model to the part of the dataset whose lines did not have any specific

Table 3.11 – *The likelihood and frequency of some specific words*

Word	Frequency in Dense class	Frequency in Non_Dense class	likelihood [2]	likelihood [3]
%	47,147	138,149	-1.6851	-1.4624
ml	137	655	-2.1689	-1.9462
ppm	4,762	48,380	-2.9282	-2.7055
ohm-m	48	150	-1.7355	-1.5128

[2]Position-based Naïve Bayes classifier [3] Naïve Bayes classifier

labels, and we assigned "IDK" labels to them (about 15,498 lines), which 7,941 of them were assigned to *Dense* class and 7,557 of them to *Non_Dense* class.

Since our experiments highlight that, in general, the Naïve Bayes classifier has high performance, we can conclude that some specific words or characters significantly impact assigning a label to a line. For example, by scanning different documents, we see mostly the cells in tables are numerical values containing percentages or units. We expect that the character "%" or units such as ml, ppm, and ohmm should have a higher frequency in dense lines with a higher likelihood for this class, as seen in Table 3.11. The negative value for likelihood is a marker for Non_Dense class.

3.6 . Semantic Analysis

Now that we have the final model, adding the predicted layout label as a new element to our data model as an engine-independent property is possible. Sometimes we need to extract information only from the paragraphs and dense_part of the texts not from the tables and images. To this aim, we can easily filter the lines based on their density for further analysis. This additional analysis can be any NLP applications on this filtered part of the texts of documents, such as entity extraction, information retrieval, etc. In this section, we explain our method to extract the information that we want. We describe an ad hoc approach to show how we can extract specific entities and relations in the OCRANA framework.

As it is already mentioned in section 3.5.1, we are able to apply specific labeling functions on top of POS tags (a rule-based chunker on POS tags) to label noun or verb phrases. The same idea can be applied to label more information in the text, such as interval, cardinal digits, or . . .; The example of this extended grammar can be seen in Figure 3.20.

```
grammar = r"""
      Interval:{<FW|CD|HYPH|NN>+<_SP?>*<HYPH|SYM|IN|TO>+<HYPH|FW|CD?>+<_SP?>*<NNP|NN|NP>+}
          {<FW|CD>+<_SP?>*<IN>+<FW|CD?>*<NN.*>+}
      CD: {<_SP?>*<CD>+<_SP?>?}
      NP: {<DT>*<NN.*>+<JJ>+<NN.*>*<NN.*>}
          {<DT>*<NN.*>+<NN.*>*<NN.*>}
          {<PRP\$>+<JJ>+<NN?>}
          {<PRP\$>+<NN?>}
          {<JJ>+<NNS>}
          {<DT|PP\$>?<JJ>*<NN>*<DT?>}
          {<NNP>+<DT?>*<_SP?>}
          {<RB?>*<JJ.?>*<.?>*<JJ,?>*<.?>*<NN.*>}
          {<LS?>*<.?>*<NN?>}
      VB: {<VB.*>+<VB.*>}
          {<VB.*>+<NN|NNP?>+<VB.*>}
          {<VB.*>}

    """
```

Figure 3.20 – *an example of extended grammar used to tag information such as cardinal digits or wellbore depth intervals*

Thanks to well-structured data model elements, we can easily extract domain-specific relation-based information, such as the interval depth of specific formations mentioned in wellbore documents. As an example of the relation-based information $\mathbf{R}(term1, term2)$, we show how the Depth intervals related to the specific formations $\mathbf{R}(Formation, DepthInterval)$ where term1 is Formation and term2 is Depth Interval can be extracted. To extract this information, we need to follow some heuristic steps, which we detail in the following:

- **Data Input**: We need to filter the dense lines of documents stored based on our data model since our target text is the dense parts of the documents. Therefore, we make our search area to extract the considered relation smaller.
- **Matchers**: Matchers can have two different types: (1) constraint-based rules or (2) dictionary-based matchers. The constraint-based rules we defined are regex patterns to tag the noun and verbal phrases. We could extend the regex pattern to annotate the intervals and cardinal digits (See Figure 3.20. These constraint-based rules can be extended for different kinds of relation-based information to tag some other types of patterns. The dictionary-based matchers match an existing term in the dictionary with the same term in the input texts. For example, for $\mathbf{R}(Formation, DepthInterval)$, the Formation terms come from a dictionary matcher provided by the user, and the Depth Intervals terms come from a constraint-based rule (Regex pattern).
- **Candidate Generation**: As the result of our matchers, we have the lines which contain Formations or/and Depth Intervals. All of these lines are our potential candidates, and we keep the index of lines as our potential candidates.
- **Window Span**: We need to limit the number of candidates to find the terms with the defined relationship. So, we define a window span size, and we predict one specific relationship can occur in that window span size. For example, in geological reports, mostly when we have the name of the

Formation, the depth interval related to that Formation should appear in the same line, following line, or maximum in the following two lines. We considered a span window to consider two terms that have a relationship. The user can define the window of span and contains the window-span lines after each term1 to consider term2 as a candidate. For example, we could consider three as our window span, assigning the depth intervals to the current formation candidate if any depth interval line candidate accrues in the following three lines.

These heuristic steps are very similar to the steps to extract the information from the Fonduer approach presented by the team of Christofer Re at Stanford [59]. The main difference is in Fonduer; originally, all matches from the Matchers step were considered related. Then the initialization space for finding the relations in the candidate generation step is huge for a big document with several pages. Because, based on this model, a value on the last page can have a relationship with an entity mentioned on the first page, even though by writing some optional Throttlers, users can filter rules to reduce the number of candidates that are already initialized together. However, here, we can only initialize a relation among some matches, and then we filter. If we have M mention matches for term1 and N mention matches for term2, in founder, first, we need to store $M \times N$ candidates, but here we need to keep $M + N$ candidates. In both cases, the id of a sentence or line is stored alongside the match mentions. The Trottlers in Fonduer method is very similar to our window of span.

Example 8. *Let's consider that as a user, we are looking for the relations of* $\mathbf{R}(term1, term2)$ *in the specific document D, where term1 is Formation which a list of them is provided by the user, and term2 is depth interval which can be found based on a regex pattern written by the user. Document D is stored based on OCRANA's elements. Then our matcher to find the formations is a dictionary-based matcher, and the depth interval is tagged "Interval" based on the regex pattern, which is already defined. The list of lines that contain any of these two pieces of information (formation or/and depth interval) for the page shown in Figure 3.21 is gathered as our generated candidates, as an example of those collected candidates can be seen in Table 3.12 and Table 3.13, respectively.*

Table 3.12 – *An example of extracted Formation candidates in OCRANA*

idx	lineindex	Line	Layout_label	Formation
0	657	Maureen Formation 5070-5353 ft MDRKB...	Dense	Maureen
1	658	Initially lithologically similar to..	Dense	Lista
2	659	Mureen Formation was Provisionally..	Dense	Mureen
3	671	Ekofisk Formation 5335-6375 ft MDRKB ...	Dense	Ekofisk
4	672	The Ekofisk comprised exclusively limestone...	Dense	Ekofisk

Table 3.13 – *An example of extracted depth interval candidates in OCRANA*

idx	lineindex	Line	Layout_label	Interval
0	657	Maureen Formation 5070-5353...	Dense	5070-5353 ft MDRKB
1	657	Maureen Formation 5070-5353...	Dense	-4988 to -5253 ft TVDSS
2	670	Rates of Penetration ranged..	Dense	28 ft/hr to 211 ft/hr
3	671	Ekofisk Formation 5335-6375	Dense	5335-6375 ft MDRKB
4	671	Ekofisk Formation 5335-6375...	Dense	-5253 to -5293 ft TVDSS
5	676	Rates of Penetration ranged...	Dense	8 ft/hr to 54 ft/hr

Example 9. *Let's consider two sets of lines $L_{term1} = \{L_1, L_2, ..., L_n\}$ and $L_{term2} = \{L_1, L_2, ..., L_m\}$ as the generated candidates belonging to both terms in the relationship $\mathbf{R}$ in document D. To assign the exact value for term1 from a set of L_{term1} to the precise value of term2 in a set of L_{term2}, we consider a window span of 3. This window span means that for each $L_i \in L_{term1}$ if there is an $L_j \in L_{term2}$, that $0 \leq index_{L_j} - index_{L_i} \leq 3$ ($index_{L_k}$ is the index of the line L_k), the value of term1 in L_{term1} is in relation with the value of term2 in L_{term2}. For example, by considering Table 3.12 and Table 3.13 as the extracted candidates for both terms of relation $\mathbf{R}(formation, depthinterval)$, the record 0 of Table 3.12 has a potential relation with records 0 and 1 of Table 3.13 (for both records: $0 \leq 657 - 657 \leq 3$).*

The original page of document D and the final extracted relation of $\mathbf{R}$ in the format of OCRANA from this page is shown in Figures 3.21 and 3.22, respectively.

Maureen Formation **5070 - 5335ft MDRKB (-4988 to -5253ft TVDSS)**

Initially lithologically similar to the overlying claystones and sandstones of the Lista Formation, the Maureen Formation was provisionally picked by an increase in the calcareous component of the claystone. The claystones were medium grey to light blue grey, soft to firm, subblocky to subplaty and slightly to moderately calcareous grading in part to marl particularly towards the base of the unit.

Sandstones continued to be very poorly cemented and were represented in cuttings by loose sand grains of clear to translucent quartz, fine grained to rarely medium, well sorted with subangular to subrounded shape and good sphericity. Where the grains were cemented, induration was very poor with a weak siliceous and occasionally calcareous cement being weakly developed.

Limestones formed minor elements within the dominantly clay /sandstone lithology, typically in stringers towards the base of the unit. The limestone was off white to, locally slightly greenish white, mudstone which was firm, blocky and with no visible porosity.

Rates of penetration ranged from 28 ft/hr to 211 ft/hr

Background gas ranged from 0.09 - 0.18 %; C1 49 - 465 ppm

Ekofisk Formation **5335 - 5375 ft MDRKB (-5253 to -5293 ft TVDSS)**

The Ekofisk comprised exclusively limestone becoming slightly argillaceous with depth. The rock was off white to slightly greenish white though it became substantially more pinkish brown towards the base of the interval. The lime mudstone was chalky in appearance, firm to very firm, blocky, with no visible porosity.

Rates of penetration ranged from 8 ft/hr to 54 ft/hr

Background gas ranged from 0.1 - 0.12 %; C1 57 - 227 ppm

Figure 3.21 – *example of the original page of reports from which we want to extract relation $\mathbf{R}(Formation, depthinterval)$*

word	Layout_Label	Formation	Interval1	Interval2
Maureen Formation 5070 - 5335ft MDRKB (-4988 to -5253ft TVDSS) \n\n	Dense	maureen	5070 - 5335 ft MDRKB	-4988 to -5253 ft TVDSS
Ekofisk Formation 5335 - 5375 ft MDRKB (-5253 to -5293 ft TVDSS) \n	Dense	ekofisk	5335 - 5375 ft MDRKB	-5293 to -5938 ft TVDSS
Tor Formation 5375 - 6020 ft MDRKB (-5293 to -5938 ft TVDSS) \n\n	Dense	tor	5375 - 6020 ft MDRKB	-5938 to -6929 ft TVDSS
Flounder Formation 6020 - 7011 ft MDRKB (-5938 to -6929 ft TVDSS) \n\n	Dense	flounder	6020 - 7011 ft MDRKB	-6990 to -7013 ft TVDSS
Herring Formation 7011 - 7072 ft MDRKB (-6929 to -6990 ft TVDSS) \n	Dense	herring	7011 - 7072 ft MDRKB	None
Plenus Marl Formation 7072 - 7095 ft MDRKB (-6990 to -7013 ft TVDSS) \n	Dense	plenus marl	7072 - 7095 ft MDRKB	None

Figure 3.22 – *Example of extracted relation* $\mathbf{R}(formation, depthinterval)$

No need to mention that to extract the entities (terms in a relation) regardless of their relation, we only need to apply the regular expression or use the external dictionaries to identify the entities.

3.7 . Conclusion

OCRANA is a framework to handle large amounts of documents and transform them into a structured format that is easy to use for further analytics. This additional analytics includes NLP applications such as entity or relation extraction. Most features related to documents' physical and logical layouts are stored in a structured format based on the OCRANA data model. One of the main goals of this chapter is to recognize the document structure of various documents. To this aim, We classified the layout structures into two main classes, Dense and Non_Dense, identifying the dense and non-dense parts of documents. Our experiments first highlight the relevance of appearing specific words and characters in each line on the final assigned labels. To have a classification model to give the labels to new upcoming documents and those parts of lines to which a labeling function could not set a label, we used a Naïve Bayes model. However, in addition to the relevance of appearing specific words in each class, we consider the significance of the positions of lines in the final assigned labels, and we proposed the position-based Naïve Bayes model. Our experiments show the performance of the position-based Naïve Bayes model to give the binary layout labels is very promising. At last, we showed how based on the semantic labeling functions. We can label more information in the OCRANA model. In this case, extracting specific information (semantic analysis) is very promising. We showed how much the named entity recognition and information extraction from unstructured formats could be done effectively and efficiently through OCRANA structured format by providing a few examples to extract this information based on the dictionaries and regex matchers. However, we still face some specific problems which need more investigation:

1. The dictionaries for specific entities (terms in a specific $\mathbf{R}$) are not inclusive and complete. The part of extracted information and relations in the OCRANA model is limited to the existing terms in the dictionaries.
2. The regex extracting the intervals is powerful in recognizing the patterns. Still, there are many similar patterns to the depth interval in the domain which are not depth intervals, e.g., records 2 and 5 of Table 3.13 are rates of penetration, not depth intervals. Still, they were extracted as interval candidates because of their similarity to the regex matcher.
3. Candidate ranking in case of overlap, even considering a window span, is another challenge that needs to be considered. For example, if we have different entities in a window span of size 3 for more than one unique Formation or Depth interval, what will be the final selected candidates in this case?

To solve these challenges, we can think of these specific solutions:

1. We need an approach to generate enriched inclusive dictionaries for domain-specific entities. In this case, we can label more entities and extract more information. To this aim, we will explore an automatic domain-specific gazetteer generation approach in the next chapter (Chapter 4). Thanks to

these gazetteers, named entity recognition and information extraction tasks in the OCRANA model will be more comprehensive.

2. For each regex matcher for specific R, we need to explore the wrong labeled entities and write a function to exclude those records, or in our case, we can ask the geologists as the experts to give us this kind of information as well. For instance, the numeric range for penetration rates seems very far from the numeric value of depth intervals. If a pattern repeatedly happens in wrongly selected candidates, we can filter that specific pattern (e.g., units of ft/hr, which does not indicate intervals).
3. To solve the problem of multiple candidacies, in our current approach, we consider those with a lower distance ($index_{L_j} - index_{L_i}$) among all valid potential candidates. However, still, we may have duplicate candidates with the same minimum distance. In this case, we can use learning to rank algorithms that leverage a rich set of features representing each candidate, including (i) candidate geospatial and (ii) document/entity coherence features.

Named entity recognition and, of course, information extraction in its continuation can be enriched by discovering an approach to generate dictionaries of domain-specific entities. We can extract more comprehensive domain-specific information in the OCRANA framework with the help of these dictionaries. Then we need an approach to generate enriched inclusive dictionaries for domain-specific entities. In this case, we can label more entities and extract more information. To this aim, we will explore an automatic approach for domain-specific gazetteer generation approach in the next section. With the objective of these gazetteers, named entity recognition and labeling the lines will be much easier.

The foundation of knowledge extraction is information extraction. The automatic extraction of information from unstructured formats using OCRANA can help construct an oil and gas knowledge base automatically. Also, in the future, we can experiment with other models for the final layout labeling classification model, such as the BERT model. We can add the positions as an external embedded layer to the BERT model. A more exhaustive benchmark can provide more insights to see whether adding position information, even in other binary classifications, can always help to have better results. Also, the next step could be exploring a more detailed classification, not just a binary classification. More detailed classes such as title, header, footnote, and paragraph in the Dense class or table and figure labels in Non_Dense class can add more value to the OCRANA framework. Also, right now, OCRANA can keep the structure of documents in which their lines are straight. We want to investigate how OCRANA can detect curved lines in the future. It will be possible by having a very well-defined training dataset of existing curved lines and with the help of a deep learning method. Also, solving the problem of multiple candidacies is demanding and essential to avoid disambiguation. In this case, the precision and accuracy of final extracted relations from unstructured documents will be higher, leading to a more reliable framework.

4 - Unsupervised NER by Automatic Generation of Domain-specific Gazetteers

4.1 . Introduction

One of the important tasks in information retrieval is named entity recognition (NER), which categorizes specified entities in texts. Many researchers are working on this domain; based on these researches, NER methods can be categorized into three main approaches: lexicon-based, rule-based, and machine learning-based [60] [61]. Also, combining these approaches is very popular as well [62] [63] [64]. When labeled data is provided, NER, which is fundamentally a classification problem, belongs to supervised machine learning [65]. Since a lot of research has been done on the general domain, the labeled data can be found for those entities. However, for domain-specific entities, many challenges still exist. Due to the critical essence of some specific domains, NER is used extensively on their data, such as biomedical data for gene, DNA, drug names, or/and disease names identification. However, the other fields, including oil and gas, do not have this privilege. Recent machine learning, such as deep neural models, has recently allowed the development of accurate NER systems. However, these techniques need a lot of manually annotated training data [60][66], which is particularly challenging in specific domains where domain experts' annotation is expensive and slow to obtain [66]. For annotating the NE corpus, open knowledge bases or dictionaries became popular [67], for example, WikiData [68] and YAGO [69] in the general domain or MeSH and CTD [1] in the biomedical domain [66]. Creating training data for NER at a large scale without human labor is possible thanks to this type of dictionaries. However, the question will remain if these types of dictionaries do not exist in a specific domain, how can we build a NER system without human efforts to provide those dictionaries? Also, the predefined dictionaries still do not contain the not very common abbreviations or most frequent typos if the corpus is noisy.

The main problems that we encounter in developing a NER system are as follows:

— Sub-surface geology domain is a very low-resource domain in which existing dictionaries, knowledge bases, or corpora are very limited. There are no comprehensive dictionaries for the entities in the domain.

— The corpus generated from the dense part of reports represented in the OCRANA model (Chapter 3) is a very noisy corpus containing many typos made originally by humans in the texts and/or made by the OCR engine. It is due to low-quality outputs while extracting the texts from scanned images

1. https://www.ncbi.nlm.nih.gov/

of documents.

— To have a NER model, we need a labeled corpus for our training set. However, we do not want to use human/expert efforts since it is expensive and time-consuming in a specific domain.

To tackle these challenges, we propose GAGNER (GAzetteer Generation for Named Entity Recognition) to automatically generate the lists of entities (gazetteers) to create an annotated named entity corpus. We use the vector word representation as the output of shallow neural networks to generate the gazetteers, each representing a specific domain-specific entity type. Researches show that Large unannotated corpora can be used to train word representations, which can then learn implicit semantic and/or grammatical (syntactic) information. [70] [71] [72].

We summarize our contributions as follows:

— We propose GAGNER, a novel unsupervised approach, to generate domain-specific gazetteers for domain-specific entities.

— GAGNER uses vector word representations to find similar words in terms of implicit semantic and/or syntactic information to generate a different group of entities in the form of gazetteers.

— GAGNER does not need external resources such as the Wikidata knowledge base to generate the gazetteers; It only uses the corpus text as its input which is a promising approach for low-resource domains.

— GAGNER can tag the words' less-known abbreviations and the wrong written forms (typos) in noisy corpora. Thanks to this ability, the generated text as the output of OCR engines that do not have high quality can be annotated.

— With the help of these domain-specific gazetteers, an annotated corpus is generated to feed a neural model to build a NER system. In other words, the final NER neural model builds a NER system by minimum resource (only using the corpus text).

GAGNER is an unsupervised model using neural networks. We use the generated result of GAGNER to train a supervised model. GAGNER result is used in a weakly supervised approach to have a neural model for NER. Without the requirement for manually generated features, neural models have produced state-of-the-art performance in NER tasks for various languages and domains. These models still need training data that has been manually annotated, but many languages and domains lack this type of training dataset [72] [73]. Since the annotation process needs human labor, it requires time and financial expenses. Unsupervised word representation may offer a solution for this issue. Large unannotated corpora can be used to train word representations, which can then learn implicit semantic and syntactic information [70] [71]. We finally annotate the corpus with generated gazetteers and build a NER system by applying a transformer model (BERT). Since there is not any benchmark of NER on the specific domain of oil and gas, we

benchmark the result with a traditional NER model (Conditional random fields (CRF)) to see the performance of a domain-specific NER system on the generated gazetteers.
This chapter is organized as follows: Section 4.2 reviews the related studies on named entity recognition and explains different word representation approaches. Section 4.3 presents a brief overview of GAGNER. Section 4.4 details the automatic generation of domain-specific gazetteers and how we evaluate their accuracy. Section 4.5 explains how we used these generated gazetteers to annotate our corpus to build a training data set for our ML model. Section 4.6 details the final NER model (BERT) learned on the generated annotated corpus and the model evaluation. Finally, Section 4.7 presents the contributions and conclusion.

4.2 . Background and Preliminaries

Named entity recognition (NER) is the process of identifying and extracting entities and information units, including names of people, places, and products. NER is a crucial NLP task and an essential component of many language pipelines, such as Q&A and information retrieval [74]. NER systems tended to use manual rule-based algorithms. However, the new systems often use supervised and semi-supervised machine learning techniques. A significant amount of annotated data is needed for supervised learning, which is very costly to gather. Then as a solution to this problem, semi-supervised and unsupervised learning are proposed by using unlabeled data. The most cutting-edge method nowadays involves using transfer learning. The NER approaches can be grouped into three main categories: (1) Traditional approaches, (2) Statistical approaches, and (3) Hybrid approaches [8] [75] [76] [77] [78] [79]. Traditional approaches mostly contain rule-based and gazetteer-based approaches (also known as dictionary-based techniques), which rely on the use of gazetteers (dictionaries) that have a list of predefined named entities. Statistical or neural network methods rely on applying machine learning methods. Therefore, sometimes these two methods are combined in a hybrid approach. Hybrid approaches combine the advantages of traditional approaches, such as gazetteer-based or rule-based approaches, with statistical or neural network methods or both. Different researchers have proposed a number of hybrid NER techniques [67] [80] [81] [82]. These techniques need a sizable quantity of training set for their supervision. Some researches indicate how we can build a NER system automatically with minimal need for supervision [80] [83] [84] [85] [86]. Word representation models from unlabeled text data have proven beneficial for many natural language tasks, including NER [67][87][88]. These word representation models are mostly based on neural networks. More recently, it has been demonstrated that vector-based word representations can capture linguistic characteristics of both a semantic and syntactic nature [70]. Word embeddings or the vector representation of a word express a word as a vector in a predetermined N-dimensional space. The closer the vectors

are, the more the two corresponding words are considered to share a similarity, syntactically or semantically [89]. Typically, a neural network that learns to predict the most likely word given a list of context words produces word embeddings [90].

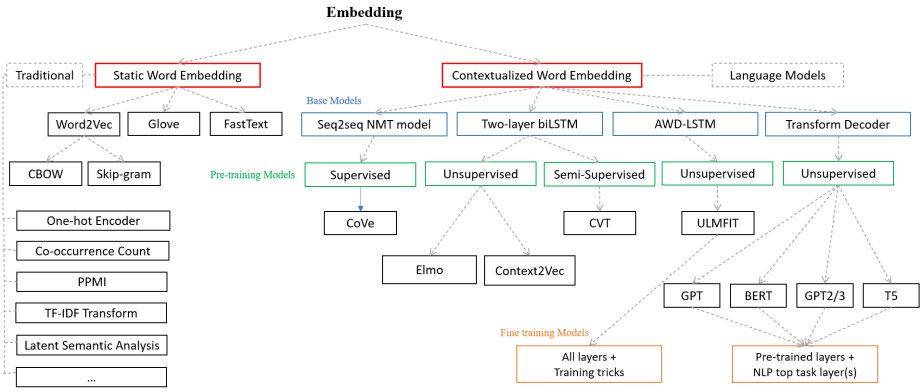


Figure 4.1 – *Word embeddings methods [91]*

Two things are always needed to create word embeddings: a text corpus and an embedding method. There are many types of embedding methods. Modern methods based on machine learning models are set to learn the word embeddings. Machine learning algorithms use the surrounding words in a corpus sentence to predict a term. These models will ultimately define the meaning of the individual words. These are machine learning models that use a self-contained data set, called a corpus, to train their models. The training data is unlabeled and supervised because the data provides the necessary context, which would ordinarily make up the labels. So, the corpus is both the training data set and the data that enables supervision. We can categorize these machine learning models depending on the factors that interest us. One main character is based on the context in which the word appears, which can have different embeddings. In this case, two main categories are (1) Static Word Representation and (2) Contextualized word representation. This categorization is interesting since it can precisely map into different main methods: (i) Shallow neural network methods, and (ii) Deep neural network methods.

A comprehensive categorization of word embedding methods is shown in Figure 4.1. The word embedding methods are categorized into two main categories: Static word embedding (shallow neural networks) and contextualized word embedding (deep neural network) methods [91]. Since we use some of these methods in our approach, briefly, we explain them in the following sections.

4.2.1 . Static Word Representation

In contrast to traditional word embedding methods (e.g., one-hot encoding, TF-IDF transforms, co-occurrence matrix, and others [92]), the static word representation can process billions of word occurrences in provided corpora. They

also can create semantically and syntactically meaningful word representations very quickly. They are called static because they generate the same vector for the exact words in different contexts. The most popular of this group of works is word2vec [93], presented by Mikolov et al. (2013). Word2vec is a neural network that creates dense word vector representations by capturing both contextual and semantic similarities. It can process substantial text corpora, build a vocabulary of potential words, and produce dense word embeddings for each generated vocabulary. Word2vec uses two different models that aim at maximizing the probability of a word given its context: the continuous bag-of-words model (**CBOW**) and the skip-gram model (**Skip-gram**) [93]. Word2vec encodes each word in a vector like an autoencoder and compares words to their nearby neighbors in the input corpus. Skip-Gram captures information about word order, while CBOW does not. The computation of the probability distribution of the context words is impractical for big datasets. For this reason, Mikolov et al. [94] introduced a negative sampling approximation (SGNS) in which the weights of random negative samples are not updating, leading to a faster model for big datasets. Another model was introduced based on a combination of matrix factorization and local context window methods called **Glove** [95] by Pennington et al. (2014). Glove (which stands for global vectors for word representation) is an unsupervised method to generate word vector representation. It analyzes word contexts and iterates on word windows across the corpus. Unlike the occurrence matrix, the word-word co-occurrence matrix indicates how frequently a specific word pair appears together. Later on, a new architecture of word2vec was presented to enrich word vectors with subword information (**fastText**) [96] by Bojanowski et al. (2016). They proposed character-level word embeddings to overcome the limitation of Word2vec and Glove approaches: Word embedding is based on a predefined dictionary which causes an inability to learn representations of rare words and out-of-vocabulary (OOV) words. FastText uses the skip-gram model and considers words as an n-gram of characters. Special boundary symbols $<$ and $>$ are inserted at the start and end of each word to differentiate prefixes and suffixes to learn a representation for each word. This allows the model to fit in with previously unknown words, often known as out-of-vocabulary terms (OOVs). FastText also enables the creation of vector representations of phrases and sentences by averaging word embedding vectors.

4.2.2 . Contextualized Word Representations

For each word, static word embeddings produce a single representation. It is a significant problem with static word embeddings which cause all senses of a polysemous word to share a single vector. To overcome this limitation, Contextualized Word Representations were proposed. Contextualized word representations, also known as contextual embeddings, are deep neural language models developed through deep learning that are customized to build models for various downstream NLP tasks. The success of this method shows that these representations reflect linguistic characteristics that are highly transferable and task-independent [97]. **CoVe**

(Salesforce [98]), **ELMo** (Allen Institute for AI [99]), **BERT** (Google [100]), **GPT-2** (OpenAI [101]), and **GPT-3** (OpenAI [102]) are the most famous contextualized word representations. ELMo (which stands for Embeddings from Language Models) is a 2-layer biLSTM model trained on a bidirectional language modeling task to generate contextual representations of each token. In contrast, Delvin et al. [100] and Radford et al. [101] proposed a new method to transfer self-attention blocks without needing to change the architecture for a specific problem. They suggest fine-tuning bidirectional transformers which have already been pre-trained for particular problems. To better understand how BERT and GPT-2/GPT-3 model works, it's better to explain transformers briefly.

Transformers are neural networks with an encoder-decoder architecture. It was used initially for machine translation from English to German by Vaswani et al. (2017)[103]. . Before transformers, RNN models were the main models for machine translation. However, transformers model a text entirely based on the attention mechanism. The main idea of this attention mechanism is to calculate the relationship between each word of an input sentence with all its words. The OpenAI GPT model represented by Radford et al. (2018) [101] suggests stacking the encoders to be able to use the architecture as a language model.

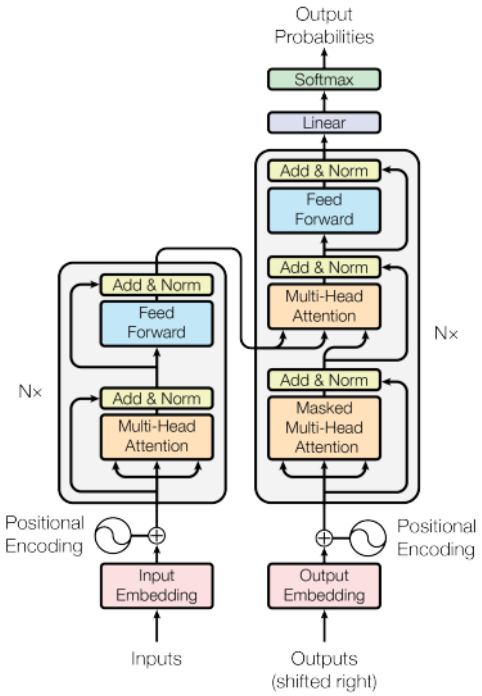


Figure 4.2 – *Transformer Architecture [103]*

As can be seen in Figure 4.2, Transformer can jointly attend to information coming from different representation sub-spaces at multiple spots thanks to multi-

head attention [103]. The obtained model generates word embeddings from the context before the word (left to right) but not the right to left context, making it faster but losing bi-directionality.

Bidirectional Encoder Representations from Transformers (BERT), a transformer-based machine learning method presented by Devlin et al. (2018) [100], "is designed to pre-train deep bidirectional representations from an unlabeled text by jointly conditioning on both left and right context in all layers" [100]. There are two steps of pre-training and fine-tuning in the BERT model. A language representation model is generated during pre-training using a significant corpus, such as the entire Wikipedia. Fine-tuning this large model to specific NLP tasks, such as NER or Q&A models, is done by exploiting the language fundamentals learned during pre-training [100] [101].

Generative Pre-trained Transformer (GPT) models presented by OpenAI were introduced first by GPT-1 [101]. To create a language model, GPT applies generative pre-training to a wide corpus of unlabeled text. This model was developed by a zero-shot approach to be applied to different NLP tasks. The ability of a model to complete a task without having already seen any examples is referred to as zero-shot learning. GPT-2 [104] was proposed a year after, and it was a subsequent improvement over GPT-1 (by adding more parameters and using a larger dataset) to learn a stronger language model. GPT-2 has an outstanding performance on zero-shot learning in zero-shot task transfer. In a zero-shot task transfer scenario, the model is given few to no examples to help it comprehend the task (also known as meta-learning). Then finally, GPT-3 [102] was proposed in 2020, one of the most effective models NLP has seen. It was trained on five distinct corpora, each with a specific weight using 175 billion parameters. The different parameter of these methods is mentioned in Table 4.1.

Table 4.1 – *Characteristics of different transformer models*

Model	# Parameters	Context window size	# Hidden layer	# Tokens	# Attention layers
BERT	110 million	512	768	30,522	12
GPT-1	117 million	768	512	40,478	12
GPT-2	1.5 billion	1,024	1,600	50,257	48
GPT-3	175 billion	2,048	12,888	50,257	96

GAGNER uses static word representation methods to generate word embedding vectors for a domain-specific corpus. We use these methods because they are lighter than contextualized word embedding methods, and we do not need many resources, time, and energy to train a model from scratch for a new corpus. Also, contextualized embedding vectors need many resources to provide us with acceptable vectors, but statistic embedding methods can perform well enough on a small corpus as well. We use different methods such as word2vec and fastText and merge the results from these methods to enrich the list of named entities related to different entity classes to have the domain-specific gazetteers. Then we use these gazetteers to annotate our corpus to have our training data set to train

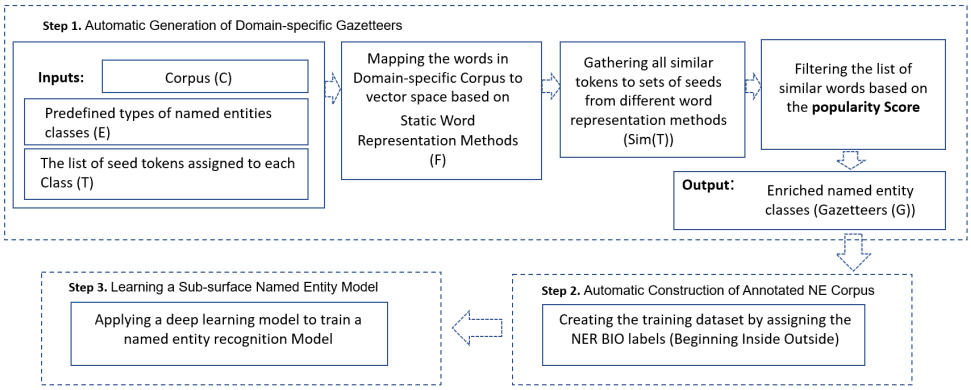


Figure 4.3 – *Proposed GAGNER Framework*

a NER model. We use contextualized word representation (BERT) to train the final model. There are many types of research to generate gazetteers automatically [83] [84] [105] [106]. A standard technique for NER includes clustering methods which means extracting named entities from the clustered groups based on context similarity. Collins et al. [83] showed that by using unlabeled data, the number of "seed" rules needed for supervision could be cut down to just 7. The method benefits from the data's natural redundancy. In many cases of named entities, the type of the named entity can be inferred from both the spelling of the name and the context in which it appears. Similarly, The difficult task of retrieving large lists of facts (such as the names of scientists or politicians) from the Web was automated by the KNOWITALL system [84]. This system bootstraps its recognition process using a small number of generalized extraction patterns and a set of seed names as input. The pipeline of the GAGNER approach is a mix of both above works [83][84]. Still, instead of bootstrapping the recognition process through a set of rules and patterns, GAGNER uses the similarity among word embedding vectors that contain semantic and syntactic information for the vocabularies through shallow neural models such as word2vec and fastText.

4.3 . GAGNER Overview

Given a Corpus (C), a set of predefined types of named entities classes $E = \{E_1; E_2; ...; E_n\}$ and the list of seed tokens assigned to these class of types $T = \{T_{e_1}; T_{e_2}; ...; T_{e_n}\}$, each T_{e_i} contains at least one token which can be classified by the type E_i and exist in the corpus ($T \subset C$). The first goal of GAGNER is to automatically generate gazetteer dictionaries that map entities to types. Each gazetteer is a defined named entity class E_i with its set of tokens G_{e_i}. This set of tokens (G_{e_i}) contains tokens similar to seed tokens T_{e_i}. In another way, it means GAGNER tries to enrich the tokens in classes related to each entity type.

Let us consider a single named entity type (E_i) and its list of seed tokens (T_{e_i})

($t_{ij} \in T_{e_i}$ where $i \in [1:n]$; $j \in [1:m]$ and preferably $m <= 20$); The goal of GAGNER is to generate a group of tokens (G_{e_i}) similar to initial seed tokens (T_{e_i}) for each entity type class (E_i). To define this similarity, we use word embeddings of initial seed tokens (T_{e_i}) by different word embedding methods (f_k), and we try to find the top most similar tokens in our corpus (C) to that initial seed tokens (t_{ij}). The collection of similar words to all words in our list of initial seed tokens ($t_{ij} \in T_{e_i}$) represents the group of similar words to G_{e_i}. We call this described process a *Similarity function* ($Sim(T)$). Figure 4.3 depicts our NER system with the help of the proposed GAGNER approach. The process of our NER system is performed in three steps which include the following details:

1. ***First step. Automatic Generation of Domain-specific Gazetteers***
   (a) Gathering a set of seeds (T) for each predefined named entity class (E); the seeds should be part of the tokens in the available domain-specific corpus (C);
   (b) Training different static word embedding methods (F) on the available domain-specific corpus;
   (c) Collecting the top-k most similar tokens in each embedding vector space for all the seeds;
   (d) Adding the new similar tokens as the output of embedding methods function in step (c) to the original set of seeds and repeat the step (c); (Steps (c) and (d) together define our similarity function process ($Sim(T)$);
   (e) Filtering the list of similar tokens based on the popularity score, which is defined based on the frequency of appearing the words appear in a different list of similar words in different word embeddings and other user-defined parameters such as the threshold for similarity rate and the iteration of repeating the similarity function;
   (f) The final sets of words after applying the popularity score in step (e) are the generated gazetteers (G) for each defined entity class (E).
2. ***Second Step. Automatic Construction of Annotated NE Corpus***
   — Generating a training data set to be used as input training data for a deep learning model
3. ***Third Step. Learning a Sub-surface Named Entity Model***
   — Applying a deep learning BERT model to train a model for a domain-specific NER

We explain these steps of the framework in detail in the upcoming sections, but first, let us clarify some concepts here.

See Example 10 where t_{31} is 'pink' and $G_{t_{e_{31}}}$ by applying the CBOW model of word2vec (f_1), and the skip-gram model of word2vec (f_2) is shown. f_k presents

a *word embedding function* that maps the vocabularies of a corpus (C) to the vectors, $h \in \mathbb{R}^d$ ($f_k : w \rightarrow h$). The function f_k has a different vector representation for different word embedding methods for the same vocabulary. In other words, a *Similarity function* ($Sim(T)$) indicates the required process to define the top L most similar tokens to the specific token (t_{ij} where $t_{ij} \in T_{e_i}$) in the vector space produced by any f_k ($Sim: T \rightarrow G$), where G is a set of a similar group of tokens ($G_{e_i} \in G$). The user can identify the value of L. Given a set of different word embedding functions ($F = \{f_1; f_2; ...; f_k\}$), the similar tokens to each specific token t_{ij} existing in the list of seed tokens (T_{e_i}) can be different for different functions f_k. The collection of all these similar tokens to our initial seed list of tokens identifies the gazetteer for the type of name entity to which the initial seed tokens belongs E_1.

Example 10. *Consider three named entity classes with the following predefined types:*
$E_1 = \{$*Age*$\}$
$E_2 = \{$*Formation*$\}$
$E_3 = \{$*Color*$\}$
Consider the list of seed tokens belonging to each class as follows:
$T_{e_1} = W_{age} = \{$*toarcian, archean, famennian, visean, anisian*$\}$ *where* $m = 5$
$T_{e_2} = W_{formation} = \{$*andrew*$\}$ *where* $m = 1$
$T_{e_3} = W_{color} = \{$*pink, grey, red, orange*$\}$ *where* $m = 4$
Consider the two different word embedding functions:
f_1: *CBOW model of word2vec*
f_2: *Skip-gram model of word2vec*

$$G_{e_i} = \{G^1_{e_i}; G^2_{e_i}, ...; G^k_{e_i}\} = \{G^1_{t_{e_{i1}}}; G^1_{t_{e_{i2}}}; ...; G^1_{t_{e_{im}}}; ...; G^k_{t_{e_{i1}}}; G^k_{t_{e_{i2}}}; ...; G^k_{t_{e_{im}}}\}$$

is a set of a similar group of tokens to the m initial seed tokens in the entity type E_i (T_{e_i}) gathered from applying the different word embedding functions, and k indicates the specific word embedding function (f_k).

Example 11. *Consider the application of the CBOW model of word2vec (f_1) on the list of seed tokens $T_{e_3} =$ {**pink, grey, red, orange**} element by element. The top 5 most similar tokens to T_{e_3}, which have a similarity of more than 90% in our corpus respectively, are:*
$G^1_{t_{e_{31}}} = \{$***pink***, *orange, pink brown, pinkish, orange brown, pale, yellow*$\}$
$G^1_{t_{e_{32}}} = \{$***grey***, *grey brown, grey green, brown grey, greyish, brown*$\}$
$G^1_{t_{e_{33}}} = \{$***red***, *reddish, red brown, black opaque, chocolate, purple*$\}$
$G^1_{t_{e_{34}}} = \{$***orange***, *pink, yellow brown, orange red, orange brown, red brown*$\}$
Consider the skip-gram model of word2vec (f_2) on the list of seed tokens $T_{e_3} = \{$pink, grey, red, orange$\}$ element by element, The top 5 most similar tokens to T_{e_3} which have similarity more than 90% respectively are:
$G^2_{t_{e_{31}}} = \{$***pink***, *pinky, red pink, pink purple, pinkbul, greeny pink}*
$G^2_{t_{e_{32}}} = \{$***grey***, *grey olive, greysih, grey brownish, grey dark, greyhard}*
$G^2_{t_{e_{33}}} = \{$***red***, *reddish, reddish purple, red purple, reddish pink, reddieh}*
$G^2_{t_{e_{34}}} = \{$***orange***, *orangey, orange gold, rd orange, red orange, brown orange}*
As we explained, G_{e_i} or $G_{W_{age}}$ is the gathering of all generated sets from different word embedding methods:
$G_{e_i} = \{G^1_{t_{e_{31}}}; G^1_{t_{e_{32}}}; G^1_{t_{e_{33}}}; G^1_{t_{e_{34}}}; G^2_{t_{e_{31}}}; G^2_{t_{e_{32}}}; G^2_{t_{e_{33}}}; G^2_{t_{e_{34}}}\}$*; Then*

G_{e_i} = {*pink, orange, pink brown, pinkish, orange brown, pale, yellow, grey, grey brown, grey green, brown grey, greyish, brown, red, reddish, red brown, black opaque, chocolate, purple, yellow brown, orange red, pinky, red pink, pink purple, pinkbul, greeny pink, grey olive, grey brownish, grey dark, greyhard, reddish purple, red purple, reddish pink, reddieh*
}
Then from 4 initial words in entity type color, we were able to generate 34 terms belonging to the color entity, and we are sure all of them appeared in our corpus.

The process described in Example 11 can be repeated in more iterations. It means that when we have $G^1_{t_{e_{31}}}$, this time, we run the same process by considering the $G^1_{t_{e_{31}}}$ as the new seed list (T'_{e_3}) with the new number of seed tokens (m'), then we try to find the most similar tokens to the new seed of the list. An example can be seen in Example 12. This iteration can be repeated for every $G^k_{t_{e_{3i}}}$ generated in the first iteration from T_{e_3} as the new seed list (T'_{e_3}). We set a threshold of 90% similarity to choose our most similar tokens. For more accuracy in the found similarity, this threshold can be increased.

Example 12. *Consider the CBOW model of word2vec (f_1) on the new list of seed tokens T'_{e_3} = {pink, orange, pink brown, pinkish, orange brown, pale, yellow} with $m' = 7$ element by element, The top 5 most similar tokens to T'_{e_3} which have similarity more than 90% in our corpus respectively are:*

$G'^1_{t_{e_{31'}}}$ = {***pink***, *orange, pink brown, pinkish, orange brown, pale, yellow*}

$G'^1_{t_{e_{32'}}}$ = {***orange***, *pink, yellow brown, orange red, orange brown, red brown*}

$G'^1_{t_{e_{33'}}}$ = {***pink brown***, *pink, yellow orange, orange, pinkish, orange brown*}

$G'^1_{t_{e_{34'}}}$ = {***pinkish***, *pale, bluish, pink, purple, off white*}

$G'^1_{t_{e_{35'}}}$ = {***orange brown***, *purplish, yellow brown, red brown, blue grey, coloured*}

$G'^1_{t_{e_{36'}}}$ = {***pale***, *pinkish, yellowish, buff, pink, bluish*}

$G'^1_{t_{e_{37'}}}$ = {***yellow***, *bright, spotty, yellowish, blue white, orange*}.

If we repeat the process for all new $G^k_{t_{e_{3i}}}$ to find the top 5 similar words to all the words in those sets, we will end up with a list of words in which some of them appear in more than one $G^k_{t_{e_{3i}}}$ and $G'^k_{t_{e_{3i'}}}$. Then we create a dictionary of all words appearing in both $G^k_{t_{e_{3i}}}$ and $G'^k_{t_{e_{3i'}}}$ in different iterations, and we keep the frequency of appearance of the words by applying different word embedding functions.

Example 13. *Consider applying the CBOW model of word2vec (f_1) element by element on the original list of seed tokens T_{e_3} = {pink, grey, red, orange} and iterate the method by applying the same CBOW model, another time on all the new set of similar words, the final entities of color derived from this method $G^1_{e_3}$ = $\{G^1_{t_{e_{3m}}}, G'^1_{t_{e_{3m'}}}\}$ contains 68 words. A sample of the name entities and their frequencies can be seen in Table 4.2.*

Table 4.2 – *Frequency of final color entities extracted by CBOW model*

Color Name Entity	Frequency	Color Name Entity	Frequency
pink	45	pink brown	2
grey	40	pinkish	4
red	41	orange brown	8
orange	46	pale	3

By applying the different word embedding functions (f_k), the similar tokens to each specific token t_{ij} existing in the list of seed tokens (T_{e_i}) will be merged to have similar final tokens for different entities (E_i). However, we need to be sure that all final tokens in E_i are correctly assigned to each named entity class (E_i). To this aim, we filter the generated set of words based on *popularity score* to prune the final class of E_i, which is generated by merging all the different groups of words generated by different embedding methods ($G^k_{e_i}$) related to a specific class of E_i. This *popularity score* is defined based on the frequency of words appearing in different lists of similar words in different word embedding functions.

4.4 . Automatic Generation of Domain-specific Gazetteers

The first step of the GAGNER approach is Gazetteers generation, whose details are explained in the following, and the whole process is done on the sub-surface geology corpus.

4.4.1 . Sub-Surface Geology Corpus (C)

We must create a corpus for NLP downstream tasks. Most of the geological information in oil and gas domains are image or scanned version documents; we first needed to use an OCR engine to extract the texts (See Chapter 3). Since the quality of these documents are not high, and the accuracy of the OCR engine models is not 100 %, we finally end up with a very noisy corpus. Our proposed model for NER is consistent with very noisy corpora. This corpus for our research is the extraction of texts from 3400 heterogeneous unstructured documents. The texts are extracted as the output of applying the Google Vision OCR engine on these documents, and after that, the image and tables inside the documents are removed. The remaining parts of the documents (the *Dense* part of the documents) were considered a sub-surface geological corpus. This corpus contains 633,337 sentences. Spacy [107] and NLTK [108] libraries are used for sentence identification, tokenization, and tagging.

4.4.2 . Collecting a Set of Seeds

Most existing approaches to NER use supervised ML models and techniques to train and generate a model for classifying pairs of terms as matches or non-matches. The critical drawback of such approaches is that they necessitate labeled training data sets. However, such annotated data sets are complex in real-world applications (for example, in the petroleum domain). Furthermore, due to the sensitivity of the

data, manually labeling a data set of sufficient size is often impossible. As a result, a novel unsupervised learning technique is necessary.

It is the only manual part of the whole approach that the user should do. The user provides the name of labels for predefined classes of named entities. The list of entity names and the number of seed tokens chosen by the user in this step is shown in Table 4.9.

4.4.3 . Training Different Static Word Embedding Methods

Since many entities exist beyond the scope of a specific corpus, but their aliases (alternative names) may match, blindly using the predefined dictionary can introduce false-positive labels in dictionary matching. For example, when the dictionary has a character name like "Charlie Brown" and its alias "Brown," several "Brown" words would be incorrectly tagged as people. The dictionary should only include the entities in the specific corpus to ensure high precision while guaranteeing fair coverage [66]. Also, most predefined dictionaries do not contain the frequent wrong written format of a word and the not well-known abbreviations of some words. We need to create corpus-related dictionaries by adding entities that appear in the given corpus. These entities can be names, aliases, wrong written forms of words, or abbreviations. We expect the NER model trained on such dictionaries to have higher precision and a good recall than that trained on the predefined, mostly small dictionaries. To do so, we use these small predefined dictionaries as our seed list and try to enrich them. We apply different static word embedding methods to generate different vector embedding spaces of words in our domain. Then, we gather all the most related words from these different embedding spaces to our original seed list of entities, and we again use this new list to find the most similar words to them in all vector embedding spaces we have.
The static word embedding methods we used here are fastText (Skip-gram subword information) and word2vec (CBOW and skip-gram) methods. We could apply more static word embedding methods to the set of seed tokens here, such as GLOVE [95]. However, we decided to test our model using these three word embedding methods. In Word2Vec and fastText, as explained in section 4.2.1, the most similar tokens to the set of seeds are gathered item by item or based on the similarity to the whole set of seed tokens. Based on the different architecture of each word embedding method, the similar final tokens to the set of seed tokens, are different. To train the word embedding representation based on continuous bag of words (CBOW) and skip-gram methods for the automatic generation of gazetteers, we used Gensim [109] and fastText [110] packages.

As we mentioned, fastText can find vectors that are out-of-vocabulary (OOV) words by adding the vectors for each of its character n-grams, but this only works if at least one of the character n-grams is included in the corpus used as the training dataset. However, for the CBOW model, it is only possible to find similar words for the existing words in the corpus, which will lead us to have original seed lists in which all of their tokens are existing words in the corpus. As you can see in

Table 4.4, there will be a limitation on the size of the seed list for the CBOW method, but we can increase the list size for the fastText method. It led us to choose the size of the original seed list based on the availability of seed tokens for the CBOW method since we want to have just one original seed list for each entity class. Also, we can find similar words to each seed token one by one (token to list), or we can find the most similar words whose vector representation is closer to all of the tokens of the seed list (list to list). In this case, all the top similar words to the seed list will be gathered by the result of the top most similar words to the whole list will be the words with the lowest average of distances to all tokens in the list. We distinguish these two approaches by adding the word *item* and *list* in word embedding methods. We collect all similar words for each type of name entity by considering a threshold and several iterations, which are explained in Section 4.3. Based on the different architecture of each word embedding method, the similar final tokens to the set of seed tokens, are different. For more clarity, we show the top 5 similar words and their similarity (1 - cosine distance of two-word representation vectors) in Table 4.3.

Table 4.3 – *Top most similar words based on different word embedding methods for the same corpus*

Model	seed token = 'wordian'		seed token = 'red'	
	Top 5 similar tokens	**Similarity**	**Top 5 similar tokens**	**Similarity**
FastText	oxtordian	0.970	reddish	0.864
	fordian	0.962	reddish purple	0.862
	lerdian	0.950	red purple	0.854
	laxfordian	0.947	reddish pink	0.850
	gaurdian	0.946	reddieh	0.840
Skip-gram	wuchiapingian	0.927	reddish	0.872
	roadian	0.925	brick	0.827
	changhsingian	0.899	red brown	0.816
	capitanian	0.894	orange	0.808
	guadalupian	0.883	green	0.805
CBOW	capitanian	.862	reddish	0.833
	wuchiapingian	0.820	red brown	0.706
	imuture	0.818	black opaque	0.705
	changhsingian	0.812	chocolate	0.675
	remesella	0.807	purple	0.668

Considering the similarity value of the top 5 most similar tokens in our specific-domain corpus for two tokens in the example of Table 4.3, "wordian" as an Age entity example (specific-domain entity) and "red" as a Color entity example (general-domain entity), the distance for specific domain tokens is lower, which means higher similarity. Additionally, since a significant portion of the component character n-grams will be the same for similar words, they will typically have a high degree of similarity in the fastText model. As a result, fastText generally does better at syntactic tasks than Word2Vec models, for example, all character gram '-dian' at the end of the top 5 similar words to the word "wordian" in Table 4.3. The result of generated corpus-based tokens for each set of seeds in each word embedding method can be seen in Table 4.6.

The five chosen different static word embedding methods (f_k where $k = 6$) in GAGNER are[2]:

1. f_1: fastText_skip-gram_SI_item (fasttext_i)
2. f_2: w2v_cbow_item (cbow_i)
3. f_3: w2v_sg_item (sg_i)
4. f_4: fastText_skip-gram_SI_list (fasttext_l)
5. f_5: w2v_cbow_list (cbow_l)
6. f_6: w2v_skip-gram_list (sg_l)

The difference between methods distinguished by the words item and the list is about defining similar words to a word of a seed list or the complete seed list of words. Item means that we add similar words to each item in the seed list. The list means we add similar words to all sets of seed tokens in one go. We collect all similar words for each type of name entity by considering a threshold and several iterations, which is explained in Section 4.3. In our domain, the manually chosen seed tokens for *Era* and *Period* type contain a limited number of entities that will not exceed this list. Then we keep the original seed tokens as the final gazetteers in our process and apply our word embedding methods to the remaining five entity types. The result of generated corpus-based words for each set of seeds in each word embedding method can be seen in Table 4.6. So far, we have noticed that the seed lists of entities are critical to generating the gazetteers. However, the size of these seed lists is a very important parameter in generating the final gazetteers. Table 4.4 shows how different sizes of seed lists of any kind of named entity types have a direct relation with the final number of generated entities in the gazetteers. Here you can see this relationship for three different named entity types. In each iteration, the top 10 similar tokens with a similarity threshold higher than α (if existing) are collected and the collective tokens added to the original seed tokens are the input for the next iteration, where again top 10 similar words with a higher threshold than α is collected.

Users can choose the threshold rate and the number of iterations. However, by default, we set a similarity rate of 80% as the threshold and an iteration number of 3 for all the methods. In the first iteration, we find similar words to the chosen set of seeds for an entity type by a similarity rate of 80%. We update the set of seeds as the input for the second iteration by a gathered set of all similar words in the first iteration. This time we apply the methods on this new set by again considering a similarity rate of 80%. The same iteration will be applied to further iterations. The number of words in the final gathered set of all similar words in the third iteration is shown in Table 4.6. We keep the frequency of appearing a unique word in any set in each iteration, then the final output of this section per each

2. The names in the parenthesis are the short form of the method names

Table 4.4 – *The impact of seed list size on generated similar tokens in each method*

Named Entity	#seeds	fastText $\alpha = 0.8$			skip-gram $\alpha = 0.8$			cbow $\alpha = 0.7$		
		1th iter.	2nd iter.	3rd iter.	1th iter.	2nd iter.	3rd iter.	1th iter.	2nd iter.	3rd iter.
Age	**1**	11	44	114	11	77	423	11	87	522
	2	20	69	144	22	107	495	22	107	549
	4	42	139	273	33	149	607	29	111	554
	8	83	231	382	60	252	845	47	186	825
Lithology	**1**	11	34	63	3	15	79	4	5	15
	2	17	55	133	4	16	80	7	13	52
	4	42	168	376	6	18	82	9	15	54
	8	83	304	813	22	94	391	32	120	359
	16	168	559	1469	-	-	-	-	-	-
Color	**1**	11	51	175	7	35	136	3	17	53
	2	22	96	265	16	78	205	14	51	127
	4	22	9	175	28	120	266	42	62	150
	8	83	272	475	52	181	362	42	85	192
	16	144	479	1072	92	298	570	65	119	266
	32	272	1100	3346	-	-	-	-	-	-

Table 4.5 – *The frequency of occurrence of a sample of terms from generated groups of similar tokens through different word embedding methods in entity class Age*

AGE	fasttext_i	cbow_i	sg_i	fasttext_l	cbow_l	sg_l	sum	#Source
wordian	427	392	399	39	39	39	1335	6
famennian	394	390	417	39	39	39	1318	6
pliensbachian	24	418	177	33	26	-	678	5
barremian	-	259	32	-	-	-	291	2
votgian	101	3	-	-	-	-	104	2
asian	240	-	-	-	-	-	240	1

method is the entity and its score (a frequency-based quotient) in each entity type class. An example of that is shown in Table 4.5.

4.4.4 . Popularity Score

Here, we define a popularity score to filter the top most frequent words generated for each entity type. The final goal is to remove the tokens that can appear in each entity type class, but there is a high chance that they do not belong to that entity type class. This popularity score ($PS(t)$) is defined based on each token's frequency of appearance in specific word embedding methods ($V(t_e^K)$: $t_e \rightarrow x_{f_i}$ where $t_e \in T$ and $x \subset \mathbb{N}$ and $i \in [1:k]$), the number of the

Table 4.6 – *The number of the words for each entity type based on different word embedding methods*

Entity Type	fasttext_i	cbow_i	sg_i	fasttext_l	cbow_l	sg_l	Sum
Age	599	1107	1539	43	43	43	**604**
Formation	932	963	809	52	44	51	**2495**
Lithology	1804	69	54	56	26		**1702**
Color	1076	90	953	46	26	56	**2624**
Texture	458	106	700	38	17	38	**1109**

Table 4.7 – *Final number of the words for each entity type after filtering by popularity score (α = 30 for all entity types)*

Entity Type	#seeds	fasttext_i	cbow_i	sg_i	fasttext_l	cbow_l	sg_l	Sum
Age	**13**	129	77	126	43	43	43	**153**
Formation	**21**	93	98	104	52	44	51	**152**
Lithology	**26**	148	35	133	55	26	54	**166**
Color	**17**	188	66	184	46	44	46	**212**
Texture	**8**	118	44	131	36	17	38	**157**

Table 4.8 – *Final number of the words for each entity type after filtering by popularity score considering α = 10 for different entity types*

Entity Type	#seeds	fasttext_i	cbow_i	sg_i	fasttext_l	cbow_l	sg_l	Sum
Age	**13**	235	98	164	43	43	43	**314**
Formation	**21**	153	130	111	52	44	51	**251**
Lithology	**26**	655	36	236	56	26	56	**780**
Color	**17**	434	76	312	46	46	46	**578**
Texture	**8**	166	47	165	38	16	38	**274**

unique time that a word appears among these methods ($P(t_{f_i})$).

$$P(t_{f_i}) = \begin{cases} 1 & \text{if } \exists t \in f_i \ \& \ i \in [1:k] \\ 0 & \text{if } \exists t \notin f_i \end{cases}$$

and the popularity score can be defined through the following equations:

$$V'(t_{f_i}) = \begin{cases} x_{f_i} & \text{if } x_{f_i} \geq \alpha \ \& \ \alpha \text{ is a threshold} \\ 0 & \text{if } x_{f_i} < \alpha \end{cases}$$

$$PS(t) = \begin{cases} \sum_{i=1}^{k} V'(t_{f_i}) & \text{if } \#Source = \sum_{i=1}^{k} P(t_{f_i}) > 1 \\ 0 & \text{if } \#Source \leq 1 \end{cases}$$

Example 14. *Considering k=6, for token t_{e_i} = wordian, $V(t_e^K)$ indicates the frequency of term t_{e_i} in different similarity groups of tokens generated through different word embedding methods (f_k). Then for our six different word embedding methods where t_{e_i} is wordian , we have: $V((t_e^1)) =$ 427, $V((t_e^2)) =$ 392, $V((t_e^3)) =$ 399, $V((t_e^4)) =$ 39, $V((t_e^5)) =$ 39 and $V((t_e^6)) =$ 39.*
Also, $\#Source = \sum_{i=1}^{k} P(t_{f_i}) = 6$. Other examples of different tokens can be seen in Table 4.5.

Also, the value for α can be modified with a user. A simple way is to consider the average of frequencies in a class as the threshold value. In other words, we consider the tokens as the final similar tokens to our seed list, which appear in at least two similarity functions. Their occurrence frequency in each method should be higher than a threshold (mean frequency) for that specific word embedding method.

If $PS(t)$ for each word has a value bigger than zero, that word will be part of the final generated gazetteers. The number of words for five types of entities for

Table 4.9 – *Geological entity type and the number of the seed list*

Entity Names	#Seeds	Entity Description
Era	14	a partition of an eon into smaller units of time in geologic time.
Period	22	one of the divisions of geologic time that allows rock cross-referencing.
Age	13	one of the geologic time divisions that splits an epoch into smaller chunks.
Formation	21	a body of rock with a predefined set of physical characteristics.
Lithology	26	the general characteristics of sediments, rocks, and rock types.
Color	17	a particular quality of some rocks that is consistently noted.
Texture	8	the composition of rocks in terms of their constituent materials.

Table 4.10 – *Ambiguity percentage between the tokens of each gazetteer*

Entity Class	Era	Period	Age	Formation	Color	Lithology	Texture
Era	100	0	0	0	0	0	0
Period	0	100	0	0	0	0	0
Age	0	0	100	0	0	0	0
Formation	0	0	0	100	0	1.05	0
Color	0	0	0	0	100	0.26	1.37
Lithology	0	0	0	0.26	0	100	0.31
Texture	0	0	0	0	1.37	0.31	100

each gazetteer can be seen in Table 4.7 after applying the popularity score. Also, we need to check the overlap of the gazetteers' different tokens with each other to be sure that each token appears in exactly one gazetteer. We want to be sure that there is no ambiguity among generated gazetteers. As shown in Table 4.10, only three of the gazetteers have a slight overlap; when we check the words appearing in these gazetteers, as a human, even we cannot apply the correct entity type; for example, the shared tokens in both lithology and texture gazetteers are: 'silt fine', 'silt vf'. V here stands for very, f stands for fine, and silt is a kind of formation; these tokens can be labeled both in lithology or texture. Since the amount of overlap is very low (1%, 0.2%, and 0.1%), a way to decrease this overlap even to a smaller value is to try different α values as the threshold to calculate $V'(t_{f_k})$.

4.4.5 . Implementation and Setup

The experiment is implemented in Python 3.6.7. To train the word embedding representation based on continuous bag of words (CBOW) and skip-gram methods for the automatic generation of gazetteers, we used Gensim [109] and fastText [110]. The common hyperparameters for training the fastText and CBOW models are as follows: vector_size (dimensionality of vector embeddings), window (context window size), min_count (words with fewer occurrences should not be used), epochs (number of epochs), and alpha (initial learning rate).

In addition, fastText has these two additional parameters: min_n (minimum length of character n-grams) and max_n (maximum length of character n-grams)

The lengths of character n-grams into which each word is divided during training and looking up embeddings are controlled by the parameters min_n and max_n regulate the number of character n-grams that each word is divided into during training. If max_n is set to 0 or less than min_n, it means that no charac-

Table 4.11 – *Training Hyperparameters for static word-embedding models: fastText and Word2Vec*

Model	vector_size	window	min_count	epochs	alpha	min_n	max_n
fastText	100	5	1	0.025	5	3	6
Skip-gram	100	5	1	0.025	5	3	0
CBOW	100	5	1	0.025	5	-	-

ter n-grams are used in the model, which effectively reduces the fastText model to Word2Vec's skip-gram model. We used similar hyperparameters for both models on four core worker threads to speed up the learning process. The value that we used for our implementation can be seen in Table 4.11.

4.4.6 . Evaluation

One question which remains is how accurate are these generated gazetteers. To answer this question, we need to show that the quality of the generated gazetteers is high enough to be used for tagging the NE corpus to have a training data set for building a domain-specific NER model. Since there is no baseline to see which percentage of all existing universal entities related to one gazetteer in the corpus is generated through the GAGNER approach, we could consider an accuracy-based evaluation. Also, as we mentioned, there is no benchmark to compare the NE corpus with a manually annotated one in this domain; we need to manually calculate the accuracy for each gazetteer. Since the number of tokens in the final generated gazetteer is a lot for some entities such as Lithology and Color, and many of the entities in the final gazetteers are with typos, it is not just about checking the generated list, we need to check them in the context in original documents, which is very time-consuming. Therefore, we sampled a random set of 50 tokens from each gazetteer, and manually with the help of geologists, we compared the following classes when comparing the accuracy:

- **True (T)** if the token in the entity type class is assigned correctly to that type class.
- **True with typos (TT)** if the token in the gazetteer is assigned correctly to that type class, but it is not written correctly. The corpus is very noisy, and it is possible that the tokens inside the gazetteers are the correct entities assigned to that class but are not written correctly.
- **False (F)** if a token is in an entity type class (the generated gazetteers for that class) but does not belong to that class.
- **Ambiguous (A)** if a token is not easily distinguished whether it is assigned correctly or not because of many reasons such as ambiguous nature or typos.

The number of each category for each entity class can be seen in Table 4.12. The accuracy of the generated gazetteers is calculated based on each class type's precision, shown in the following formula. Also, The final result for the accuracy of the generated gazetteers in GAGNER is shown in Table 4.13. The total precision

is the precision of 250 random tokens from a total of 1320 tokens in all gazetteers (around 20% of all tokens).

$$\text{Strict Precision} = \frac{T + TT}{T + TT + F + A}$$

$$\text{Lenient Precision} = \frac{T + TT}{T + TT + F}$$

Table 4.12 – *Gazetteers evaluation*

α	#Sampling	Metric	Entity Type				
			Age	**Formation**	**Lithology**	**Color**	**Texture**
$\alpha = 10$	50 samples each entity type	**True**	35	32	29	36	37
		True with typos	10	13	21	14	13
		False	3	0	0	0	0
		Ambiguous	2	5	0	0	0
$\alpha = 30$	Complete evaluation	**True**	139	138	133	193	149
		True with typos	151	145	154	203	156
		False	2	7	12	5	0
		Ambiguous	0	0	0	4	1

Example 15. *Set A. contains some formation names in the Formation gazetteer having typos (or abbreviation forms), and set B. contains the proper form of those real entity names.*

$$A = \{\textit{eather, ek, wus, torn, kimmeridye, kinmeridge}\}$$

$$B = \{\textit{heather, ekofisk, wuschel, torg, kimmeridge, kimmeridge}\}$$

As can be seen, GAGNER has promising results on precision, but we cannot be sure about the high coverage of all existing named entities in the available corpus. We could enrich a set of named entities in each entity type, but we still do not know how much the final generated gazetteers for each entity type are comprehensive and inclusive.

Table 4.13 – *Accuracy of generated gazetteers- for two different generated gazetteers by applying different threshold (α) in popularity Score to finalize the gazetteers*

Entity Type	Partial sampling ($\alpha = 10$)		Complete evaluation ($\alpha = 30$)	
	Strict Precision (%)	**Lenient Precision (%)**	**Strict Precision (%)**	**Lenient Precision (%)**
Age	90.00	93.75	99.31	99.31
Formation	90.00	100	97.59	97.59
Lithology	100	100	95.99	95.99
Color	100	100	97.78	98.75
Texture	100	100	99.67	100
Total	96.00	98.75	98.07	98.33

4.5 . Automatic Construction of Annotated NE Corpus

The sub-surface geology corpus which we created contains around 634,000 sentences. We used the generated gazetteers and annotated the corpus to have a named entity annotated corpus. Different tagging schemes exist for NER, but we used the BIO entity tagging scheme in this study (sometimes called IOB). Each token in the corpus is assigned a type and a position within the entity using the BIO scheme. *B* stands for the beginning of an entity, *I* means inside, and *O* means outside of an entity. The labels for each created gazetteer are the type of entity. We have seven entity types, which increase to 14 when considering the BIO format, and other tokens with no assigned entity type receive the O (Outside) type.

Automatic Construction of Annotated NE Corpus needs the process of Gazetteer Matching. This matching can be done in two levels: single-token matching and multi-token matching. Assigning gazetteer properties to sentence tokens is known as gazetteer matching. In this process, we need to determine the entity types of each token ti in a sentence $S = \{t_1; t_2; ...; t_n\}$ and map those tokens to entity types ($\{E_1; E_2; ...; E_n\}$), given a gazetteer dictionary M. The match span S_{ij} represents positional information that encodes multi-token matches. The match spans are encoded using BIO (Beginning Inside Outside) tags, similar to the BIO tags of the CONL-2003 dataset, and we use this format to encode the NER labels. Multi-token and single-token are the two primary approaches for gazetteer matching [67]. Multi-token matching is looking for the most extended parts of the sentence that can be mapped to a type in the gazetteer dictionary G [67]. For example, given G[Light]$\rightarrow$ Outside, which means the word "light" is not specifically among the terms in any gazetteer type classes, and G[Brown]$\rightarrow$ Color, which means the term Brown belongs to gazetteer Color entity; The combination of "Outside" vocabularies with any term in any entities in Color type gazetteers (e.g., G[Light Brown]$\rightarrow$ Color) is tagged as a Color entity. For example, consider these two sentences: (1) "**Light** metal and alloys possess high strength-to-weight ratios and low density" and (2) "The dominant lithology is **light brown** sandy clay" the multi-token matcher assigns the color gazetteer type to the most extended segment "light brown".

We also use a seed list of five tokens of epochs: ['upper', 'mid', 'early', 'middle', 'lower']. We enrich them through the same method we generated gazetteers, but we clean the list manually to have a list of 21 tokens of epochs : ['earliest', 'inner', 'intra', 'middle', 'lower', 'uppermost', 'late', 'middlt', 'iddle', 'pre', 'topmost', 'latest', 'post', 'upper', 'lowermost', 'mid', 'early','late','niddle', 'mddle', 'carliest'] This time, we apply the same method as we used for color for these entity classes: Age, Formation, Era, and Period. If any of these tokens of Epochs happen before the entity from these classes, we apply the same entity label to the token from the Epoch list. For example, if we have ['upper', 'flounder'] in our tokenized sentence, the annotated text will be ['B-formation', 'I-formation'] or for ['mid', 'jurassic'], we have ['B-Period', 'I-Period'].

Searching for exact matches between any vocabulary term from a gazetteer type

is known as single-token matching. In the previous example, each term from the sentence was individually matched to the tokens in G, then "brown" was matched to the "Color" type entity class. However, to tag the terms based on the gazetteers, the multi-token matcher looks for the most extended segments of the terms. These segments in the corpus are bigram or trigram terms.

Table 4.14 – *BIO label distribution: train and test set*

Entity Label	Training set	Test set
B-Era	215	67
B-Period	15420	3852
B-Age	15384	3766
B-color	55130	13602
B-formation	31849	7879
B-Lithology	152332	38653
B-Texture	45361	11449
I-Era	18	6
I-Period	6784	1694
I-Age	12236	3009
I-color	55735	14009
I-formation	4923	1215
I-Lithology	15446	3982
I-Texture	23013	5762
TOTAL	3208213	800172

4.6 . Learning a Sub-surface Named Entity Model

In NLP, we typically deal with variable-length text sequences, such as sentences, rather than single words without context. The order of the input words is essential here, and words that come before them will help distinguish later words. Text data is an example of sequential data without a temporal component compared to time-series data, such as signal processing data. For this sort of sequential data, specific DL approaches, such as RNNs and sequence-to-sequence models [111], and lately, attention networks and transformers [103] have been shown to function better. This is also true for the NER tasks. Until recently, when self-attention networks allowed for transfer learning, which is when a pre-trained language model is fine-tuned with a labeled NER corpus, BiLSTM RNNs, along with the best CNN algorithms and conditional random field (CRF), where the state-of-the-art in NER [63] [112] [113]. Lafferty et al. (2001) [114] initially presented CRF as a framework for statistical sequence modeling. While a conventional classifier does not include "neighboring" samples when predicting a label for a single sample, a CRF can consider the context for label prediction [63]. In order to overcome label bias issues, CRFs use the conditional probability property in place of the independence assumption in models such as Hidden Markov Models (HMMs). CRFs show better performance than MEMMs (Maximum Entropy Markov Models) and HMMs in many domains, such as biomedical, voice recognition, and computational linguistic modeling [63] [115] [116]. Now that we have defined a NE corpus, we can create a NER system to extract and classify named entities. We use a transformer model

(BERT [100]) to make a NER model. While BERT only uses an encoder stack for classification, NER uses a feed-forward layer. The essential concept is Masked Language Modelling, which allows the model to predict a word by looking at the entire sentence or text [100]. This contrasts with RNNs or Transformers, which base their predictions simply on a few words of a sentence. 15% of the tokens in the normal semi-supervised Language Modelling task are masked in BERT. After that, the model is asked to identify the word for each of these tokens. However, generalization is enhanced by using a mixture of original, random, and masked words.

BERT also goes beyond semi-supervised learning by determining whether two sentences are connected and whether they are logical continuations of their predecessor. These two tasks work together to ensure that BERT generalizes successfully and captures linguistic structures. One benefit of multi-task learning is that it is a regularizer for sentence categorization and word prediction. It can prevent overfitting to a single task because the weights must balance the need for several tasks. BERT is a language representation model with two pre-training and fine-tuning stages [100]. BERT was initially developed for TensorFlow and has since been implemented into Huggingface Transformers for PyTorch [117]. There are different models for different applications for BERT, such as BertForSequenceClassification, BertForMultipleChoice, BertForQuestionAnswering, and BertForTokenClassification. All tasks require additional domain-specific fine-tuning. BERT models come in various sizes, including base and large, as well as case sensitivity (cased and uncased) and pre-training language.

As shown in Figure 4.4, the model created consists of three primary components: Preprocessing, BERT, and classifier. For the NER task, Devlin et al. used the BERT output and designed a classifier. The classifiers in this work were developed using the Feed-Forward layer and Softmax, trying to recreate the model as was described in the original paper [100]. A ten percent dropout rate is applied to the results of BERT. Given that Devlin et al. did not share the NER application's source code and that little is known about it, it is challenging to replicate the results they have provided. There is still controversy regarding how BERT accomplished its NER results on CoNLL-03 [3].

4.6.1 . Fine-tuned BERT

This study uses a softmax classification layer similar to the BERT's original paper presented by Devlin et al. (2018) [100]. First, corpus words are tokenized into wordpiece tokens. Following that, the BERT representations are learned by the BERT encoder. Finally, the classification layer produces the BIO labels for the sub-surface geology entity type classes. We use pre-trained Bert-base-uncased using the Huggingface website [4]. This model has 12-layer, 768-hidden, 12-heads,

3. https://github.com/google-research/bert/issues/223
4. https://huggingface.co/models

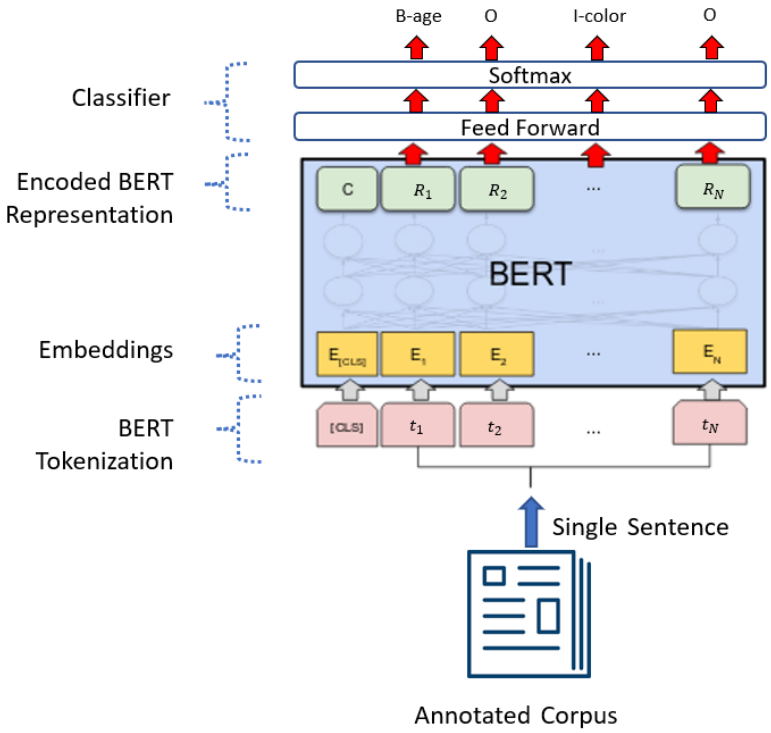


Figure 4.4 – *Simplified overview of the NER model architectures for fine-tuned BERT being developed in this work.*

and 110M parameters.

4.6.2 . Classification Layer

The current study implements fine-tuned BERT with Softmax classification, consistent with the best-performing BERT implementation by Devlin et al. (2018) [100]. This Softmax function determines the most likely class for each word in the given input [118]. The following formula represents the Softmax computation, where x_i stands for the dense layer value for NER class label i:

$$softmax(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)}$$

4.6.3 . Model Optimization

The Adam optimizer is frequently used in existing BERT experiments to update model weights [100][119]. This study uses the same optimizer from the original paper [100]. However, we also try adjusting the other hyper-parameters, such as learning rate, epochs, batch size, dropout ratio, and weight decay. Table 4.15 shows the result for the best hyper-parameters. The maximum length of an input sequence that the BERT model will accept is 512 tokens. There is no requirement to use all 512 tokens because most sentences in our corpus have significantly fewer tokens. A few sentences that are longer than 128 tokens had their sentences truncated at

Table 4.15 – *Model hyper-parameters*

Model Architecture	Parameter configuration		
	Batch size	Learning rate	epochs
BERT + Softmax	32	3e-05	3

the end. the separation token ([SEP]), the classification token ([CLS]), and the first 126 tokens ([126]) remain.

4.6.4 . Model Evaluation

The NE corpus created by generated gazetteers contains 633,337 sentences, and 184,142 out of them have at least one label of seven named entities for which a gazetteer is generated. We split the 184,142 sentences two a train and test data set by considering the ratio of 80 to 20. The train and test data set information is shown in Table 4.16. We examined the NE count and split entity types into the training/test data. The number of entities per dataset is presented in Figure 4.5. Table 4.14 shows the result of true and false predicted labels based on the different Named entity labels. This result is after applying the model BERT with the softmax classifiers.

Table 4.16 – *Training/test dataset information*

	Training	Test
# Sentences	147,314	36,828
# Tokens	3,639,957	1,330,085
# NE Labeled Tokens	433,846	181,938

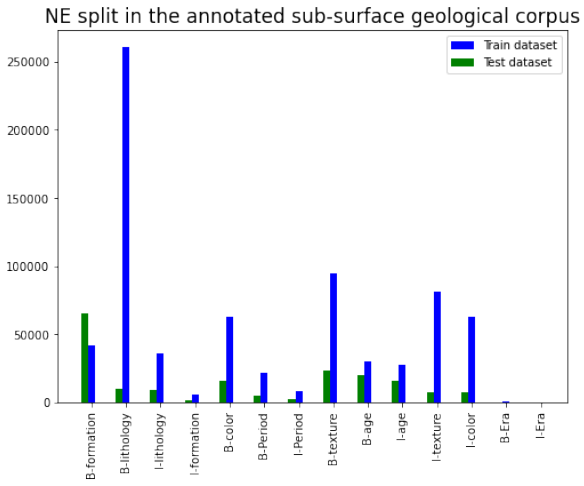


Figure 4.5 – *NE split in the annotated sub-surface geological corpus.*

The same process with the same test and training data set is done to build a NER system with a CRF classification model. The final result of the average precision, recall, and F1-score is shown in Table 4.17.

Table 4.17 – *Experiment results*

Model	precision	recall	f1-score
BERT	0.97	0.98	0.97
CRF	0.95	0.91	0.93

Table 4.18 – *BERT vs. CRF model results for test set (BIO labels)*

Entity Label	BERT Model			CRF Model		
	P	R	F1	P	R	F1
B-Era	0.98	0.96	0.97	0.93	0.89	0.91
B-Period	1.00	0.99	1.00	0.95	0.94	0.94
B-Age	0.94	0.93	0.94	0.87	0.84	0.85
B-Color	1.00	1.00	1.00	0.86	0.86	0.86
B-Formation	0.99	0.99	0.99	0.97	0.98	0.97
B-Lithology	1.00	1.00	1.00	0.98	0.99	0.98
B-Texture	1.00	1.00	1.00	0.91	0.90	0.90
I-Era	0.78	1.00	0.88	0	0	0
I-Period	0.99	1.00	0.99	0.89	0.94	0.91
I-Age	0.90	0.94	0.92	0.83	0.92	0.87
I-Color	0.99	1.00	1.00	0.87	0.89	0.88
I-Formation	0.97	0.95	0.96	0.89	0.94	0.91
I-Lithology	0.98	0.99	0.98	0.93	0.80	0.86
I-Texture	0.99	1.00	1.00	0.89	0.82	0.86
TOTAL (Macro Average)	0.97	0.98	0.97	0.84	0.83	0.84

4.7 . Conclusion

We have described the annotation of the first NER corpus with a large number of documents in the sub-surface geology domain. This corpus, with over 663,000 sentences containing over 915,000 NEs in just 184,185 sentences with 6,792,070 tokens, is one of the largest automatically annotated NER corpus available for the English language, includes a variety of text types that have been annotated for seven specific-domain entity types. To automatically annotate the NE corpus, we presented GAGNER, a novel approach for gazetteer generation that uses shallow neural network methods such as word2vec and skip-gram methods. Finally, we applied a pre-trained transformer model (BERT) on the annotated NE corpus. Since the evaluation results of generated gazetteers show high accuracy, we can be sure that the annotated NE corpus has very high accuracy. We can consider the annotated labels as high-quality manual labels. We see that the final evaluation of the NER system has a very promising result.

GAGNER is a powerful method for NER, but the type of these named entities should be words, meaning that finding named entity types containing numerical values or symbols which follow a pattern is not compatible with this method. Entity types such as dates, specific identifiers for entities, and numerical values cannot be tagged through the GAGNER approach. Then if this type of information is important for the domain, we should think about another approach. For future work, we also want to assess the recall of the created gazetteers. Since there is no baseline to compare how much the generated gazetteers cover all the entities in the corpus.

It requires manual labeling for a part of the corpus.
Also, we would like to evaluate GAGNER on non-English language datasets to show that GAGNER is generating the labels from the corpus. The foundation of our approach makes us believe that the approach can work well in other languages. We know in recent years, pre-training large neural language models, such as BERT or GPT-2/3, had impressive gains on many NLP tasks. For specialized domains like biomedical, there are many types of research to pre-train domain-specific language models dedicated to the domain. Their results show that domain-specific pre-training language models are a robust baseline for a variety of biomedical NLP tasks, such as NER, which is leading to new state-of-the-art results [120][121][122]. One idea is to discover if pre-training a language model can also bring more advantages for our specific sub-surface domain. To this aim, we define the GeoBERT model, which is presented in detail in Chapter 7.

5 - GeoBERT: NER using Domain-Specific Language Models

5.1 . Introduction

Domain-general corpora are the primary target of most pre-training language models. The widespread assumption is that domain-general language models can help even domain-specific pre-training. Numerous studies in specialized domains, such as bio-medicine, demonstrate that pre-training language models from scratch yields more significant benefits than continuous pre-training of domain-general language models for domains with large amounts of unlabeled text [120][121][122]. However, because there is a shortage of literature resources in the geological sub-surface compared to the biomedical domain, pre-training a language model from scratch means we lose a lot of the knowledge already embedded in domain-general language models, such as BERT. As mentioned in the previous chapter (Chapter 4), we face some challenges in building a NER system. Although we could build a promising NER model (GAGNER), this model held its own limitation. Also, some main problems still need to be solved.

- GAGNER is a powerful method for NER. Still, the type of named entities extracted through this approach should be words, meaning that finding named entity types containing numerical values or symbols following a pattern is not compatible with this method. These entity types include the date and specific identifiers (e.g., wellbore IDs and numerical values). Then if this kind of information is vital for the domain, we should consider another approach.
- GAGNER is helpful for the specific NLP task of NER. Suppose, we want to apply other NLP tasks, such as Q&A, in one particular domain; In that situation, a domain-specific pre-trained language model can be quite beneficial.
- The sub-surface geology domain is a very low-resource domain in which existing dictionaries, knowledge bases, or corpora are limited. There are no comprehensive and inclusive dictionaries for the entities in this domain.
- BERT is a domain-general pre-trained language model (generic BERT). The principal shortage of the generic BERT model in a specific domain is related to out-of-vocabulary words (OOV) (unseen words) and lack of precise embedding representation for them; since, in a new specific domain, lots of frequent words (presenting most of the content) are OOVs.

Pre-trained Language Models (PLMs) are refined on a downstream task like NER after being pre-trained over a large text corpus [123]. BERT (explained in Section 4.2 in the previous chapter) is a PLM offering subword representations rather than word-level representations to represent both the tokens of input text and

the output. When an unknown word (unseen word in the pre-trained BERT model) is provided to BERT, it will be divided into a number of subwords, maybe even character subwords, depending on the situation. It is the way that BERT handles unseen (OOV) words. The principal shortage of the generic BERT model in a new domain is related to the OOVs when many of the common words (most frequent words) in a new domain are OOVs, and there is a lack of exact embedding representation for them. Some models on a specific domain solve this problem by training a model from scratch on their own corpus [120] [121]. However, training the new model requires a considerable corpus and lots of GPUs and TPUs, leading to huge computational costs. When we train a model from scratch, if our domain-specific corpus is not large enough, we will lose a lot of already existing information from the generic BERT model because we will have a new set of vocabularies. Although some other models alter the pre-trained model by using it as the initial model for a new domain to tune the word embedding, they nevertheless employ the same vocabulary as the original BERT model. They initialize the weights of the generic BERT model to train a model from scratch; The issue with this model is that most domain-specific terms are still OOV, which causes sub-optimal performance on downstream tasks. This is because the original vocabulary only includes some vocabulary of a specific domain. One option to solve this problem is to integrate the most common words from domain-specific corpus into the BERT vocabulary dictionary and reuse the pre-trained model's weights to integrate the vocabulary embeddings from the new domain to the generic BERT model. Compared with pre-training a language model from scratch, it results in less computation and training data. One problem with this concept is that the original BERT vocabularies' polysemy terms, which have various meanings in other domains, nonetheless have the exact representation. Then, to overcome this difficulty, a method that enables interaction between the generic BERT model and the extension model for domain-specific words (extension module) must be used.

exBERT (extend BERT) [124], for the first time, challenges the same idea in the biomedical domain, which was presented by Tai et al. (2020). However, exBERT model was pre-trained on a large number of biomedicine resources, but as we mentioned, the geological domain is a limited source domain. In this chapter, with only a limited number of domain-specific sources, we want to challenge the same idea and adapt the generic BERT model to a specific domain with a limited number of sources. We want to see whether a limited source domain, such as the sub-surface geology domain, can also benefit from extending the pre-trained model with new vocabularies. We demonstrate this by adopting brand-new, additive vocabulary (out-of-vocabulary words, or OOVs) from domain-specific sources along with an extension module to modify the initial BERT embedding of a generic vocabulary to fit the new domain [124]. Our experiments demonstrate improvements over the generic BERT model as we evaluate our domain-specific BERT-variant language model (GeoBERT) for a downstream task: NER. Our findings indicate that this

approach is consistent with using small domain-specific sources. According to our knowledge, this approach has never been used before in the geology, oil and gas domain, or sub-surface domain, let alone on limited resources. We summarize our contributions as follows:

- Presenting GeoBERT, a domain-specific BERT-variant language model for the sub-surface geology domain.
- Adapting the generic BERT model to a specific domain with limited sources.
- Demonstrating that the generic BERT model can benefit from integrating an extension module to solve the unseen vocabulary (OOV) of specific domains in the limited-source sub-surface domain.

The chapter is organized as follows: Section 5.2 reviews the related studies on domain-general and domain-specific language models. Section 5.3 presents a brief overview of GeoBERT. Section 5.4 details the building blocks of GeoBERT. Section 5.5.2 illustrates the results of our experiments. Finally, Section 5.7 presents the conclusion and discusses future steps.

5.2 . Related Work

The majority of NER systems use pre-trained domain-general language models, like BERT, which means that their architecture is based on a language (e,g., English, french and others) rather than a narrowly defined domain. In many natural language processing applications, these pre-trained models have shown substantial effectiveness; however, they suffer from a domain shift in specific domains. Then, fine-tuning can be challenging as well. SciBERT [120], a popular BERT variant trained mostly on biomedical literature. It is an entirely new model trained from scratch with new vocabularies. We should keep it in mind that when we train a model from scratch, if our domain-specific corpus is not large enough, we will loss a lot of already existing information from the generic BERT model. Other models use the same vocabulary as the original BERT model even if they modify the pre-trained model by employing it as the initial model for the new domain. They initialize the weights of the generic BERT model in order to train a model from scratch; the BioBERT model [121], which uses this technique, is a well-known BERT variant model in the biomedical domain. The issue with this model is that most biomedical terms (domain-specific terms) are still OOV, which causes sub-optimal performance on downstream tasks when using the pre-trained model as is. This is because the original vocabulary may not be proper for a specific domain. One option to solve this problem is to integrate the most common words from the domain-specific corpus into the BERT vocabulary dictionary and reuse the pre-trained model's weights to integrate the vocabulary embeddings from the new domain to the generic BERT model. This will result in less computation and training data. One problem with this concept is that the original BERT vocabularies' polysemy terms, which have various meanings in other domains, nonetheless have the exact representation in

another domain. Then, a method that enables interaction between the generic BERT model and the extension module must be used to overcome this difficulty. exBERT (extend BERT) [124] challenges the same idea in the biomedical domain. In their search, they ran into two similar issues [124]: (1) The pre-trained generic BERT model does not know how the extension vocabularies are embedded; (2) For a variety of reasons, including differing sentence structures, formality, intent, and . . ., the distribution of token representation in the original vocabulary may face a change from the general domain to the specific domain. For instance, the same word may have distinct meanings when used in contexts of different domains. They used a weighted combination mechanism in their solution, which enables cooperation to address these problems.

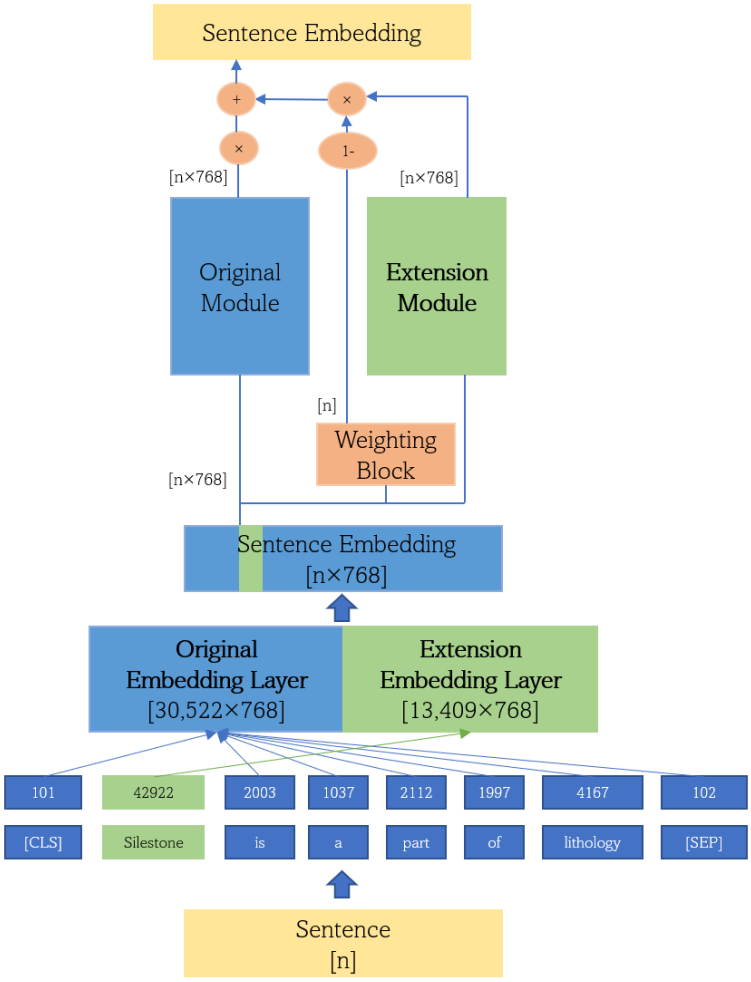


Figure 5.1 – *GeoBERT architecture*

5.3 . GeoBERT Overview

GeoBERT is a pre-trained language model, a domain-specific BERT-variant language model for the oil and gas domain. The idea is to adopt a generic BERT model to a new domain by injecting the out-of-vocabulary words into the already trained model. By using the byte-pair encoding (BPE) or variants such as WordPiece, neural language models create a vocabulary from subword fragments to solve the issue of out-of-vocabulary words [125] [126] [127]. Specific domains contain many in-domain terms, which may be divided into sub-word pieces (e.g., 'siltstone' is tokenized into ['si', '##lts', '##tone'] in the original BERT model). However, since 'siltstone' is part of the most frequent words in the sub-surface domain, having a predefined embedding vector for this word seems to enable more meaningful tokenization of input text, especially if more domain-specific terms are OOV words. Therefore, the expectation is that GeoBERT will perform better on NLP tasks in the sub-surface domain by adding the extension vocabulary and accompanying embedding layer. The leading architecture for adding these new vocabularies can be seen in Figure 5.1.

GeoBERT has borrowed its architecture from exBERT [124], which includes the original module of the generic BERT model and an extension module which adds the new domain vocabulary to the original vocabulary of the BERT model. exBERT challenges the same idea in the biomedical domains, which contain many text resources. They argue that the same word may take on distinct meanings when used in various contexts. To address these issues, they proposed a weighted combination mechanism that was applied in their method that allows cooperation (Figure 5.1). For instance, the word Silestone is considered as one vocabulary in the GeoBERT, while in BERT original model is tokenized into three tokens. Based on this architecture, we have an extended embedding layer that is added to the original module. We need to pre-train the new extended vocabularies and merge them with original embeddings of vocabularies from the original model. Then the model must be pre-trained again, considering the newly injected oov words. We explain this process more in section 5.5. To train a domain-specific model based on exBERT architecture, we need to follow two steps: (1) Adding Extension Embedding Module; (2) Pre-training process. More details of these two main parts are explained in the following.

5.4 . Extension Embedding Module

Some shortages of the BERT model in a specific domain are related to OOV vocabularies and lack of precise embedding representation for them in a new domain when lots of frequent words are OOV vocabularies. as we mentioned in related work, there are some models on a specific domain which solve this problem by training a model from scratch on their own corpus. This training of the new model needs a considerable corpus and lots of GPUs and TPUs, which lead to substantial computational costs. The other idea can be adding the most frequent words on our corpus to the BERT dictionary and reusing the original pre-trained model's weights to add the new domain's vocabulary embeddings to the original BERT model. This will cause reduction in required computation and training data. This idea will cause a reduction in required computation and training data. Also, the issue of polysemy words in original BERT vocabularies will remain that similar words with different meanings in another domain still have the exact representation. Then, in order to overcome this difficulty, a method that enables interaction between the original BERT model and extension module must be used.

To better understand why handling OOV words is so important through the extension embedding module, it's better to answer these questions :

— **How a human brain naturally can guess a masked word from a sentence? Where do we put our attention when we as humans read a text?**

For sure, as humans, our brains become certain about the meaning of a word based on the context if the word is ambiguous or not familiar to us. One known way to calculate this uncertainty of each word is by calculation by Shannon entropy [128]. Entropy is the expected value of the variable's self-information. Then as a human, even guessing a mask word of a sentence can be difficult if we are unfamiliar with the domain. e.g., a non-native English speaker Ph.D. student who is spending lots of time reading papers suddenly sees a flyer that is torn from the middle of this sentence: Eat some arti .... Her brain may fill the blank part with the word "arti-cles"; However, another person can quickly and easily fill the missed part of the words with "arti-chokes". There are two main reasons for this behavior: 1) being biased based on her previous knowledge of her expertise, and 2) lack of knowledge about all the names of edible plants in English. Self-attention-based language models do the same behavior. They try to get the essential sub-information tokens in the public domain that they have trained on and build a dictionary of known tokens to guess the vector of out-of-vocabulary words (unseen words in their training process) based on their sub-tokens. Since it keeps the attention of all tokens in the context of an input (a sentence, paragraph, or chunk of speech), this method can be helpful if we have a small amount of UNK (unknown) words. It's like you have a UNK word in a sentence. There's

not any pre-trained embedding representation for that UNK word but based on the embedding of the other known words in the context plus the known sub-token parts of that word from the dictionary, the model can assign a vector initially to the OOV word. Then this embedding representation can be improved in the NLP downstream tasks. But what if there are lots of OOV words? Many OOV words mean we are in domain-specific literature, and our model suffers from a lack of knowledge. Then generic domain such as the BERT model has a bias to guess towards the OOV words (precisely like the artichoke-article example), even if some part of word information can be found in the dictionary (e.g., arti).

— **How our brain naturally understands the meaning of a new word in a text (or understand an unknown meaning of a known word)?**

The self-attention mechanism is beneficial for encoding multiple meanings of ambiguous words This mechanism works like the human brain to understand the meaning of the mask words. For instance, consider the following sentence from the beginning of a paragraph in a random paper [129]: "It combines two crucial techniques to solve the problems of attention...". When I reach this part of the sentence, my brain can conclude that this paragraph probably talks about attention deficiency based on my previous knowledge. Then my brain guesses that the domain of this paragraph is psychology or, more generally, the mental health domain. The sentence will continue: "It combines two crucial techniques to solve the problems of attention and memory..." [129]. By adding more known vocabulary, I know memory problems are probably related to post-trauma syndrome, some neurodevelopmental syndromes, or brain damage. How do I know this? Again, based on my previous knowledge. This prior knowledge shaped my training data to predict the context and the masked words before reading the rest of the sentence or paragraph here. If you ask another person, she may or may not has the same idea. However, most people have similar opinions since their common and public knowledge are mostly the same (the same education systems, the base public knowledge), and probably now this paragraph talks about memory loss in health domain. As you noticed, the calculation of certainty of what we guess depends exactly on what we have already known (as it is always mentioned in information theory). The text continues as: "It combines two crucial techniques to solve the problems of attention and memory allocation..." [129], by appearing "allocation" in the term "memory allocation" (notice "memory allocation", not just the word "allocation"), the whole amount of certainty about even the topic of the paragraph changes in my brain. This paragraph probably speaks now about the issues in the computational tasks to use memory more efficiently (generally, computer science domain). The words such as attention, memory, and allocation are not OOV words for any educated person. They are not even domain-specific

words, but they need representations that show the certainty of all different contexts in which they can appear. Suppose we even miss any domain that can appear. In that case, it means we lose some information because of a lack of knowledge. For example, if an ex-ballerina in her 90s reads this paragraph probably doesn't have any idea, even after memory allocation terms, that this paragraph might be talking about computer research. This might happen because of her lack of knowledge in this domain; then, she may end up continuing to read the paragraph with her biased knowledge that it's an article about the health domain. Even for the known vocabularies without enough examples of a different existing domain, a representation of a known word cannot be its best representation when it comes to a specific domain. For instance, the word "Platform" encodes multiple meanings in different contexts, such as offshore platform, computing platform, train platform, hosting platform, car platform, weapons platform, concert platform, and chunky platform. Through self-attention, language models can distinguish a word's correct meaning more clearly. However, if there are lots of OOV for the language model, it's not clear that the model should put its attention based on the context of the text. The context of a text will remain ambiguous in two situations: 1) there are lots of OOV vocabularies related to a specific domain. 2) we are in a specific domain in which known vocabularies in the dictionary have other meanings which we do not have any previous knowledge about, and it can misguide us (The example of 90 years old ex-ballerina).

5.4.1 . Limited Source Corpus

To have the extension vocabulary, we need a domain-specific corpus. The sub-surface geology corpus, which we prepared in section 4.2 is very noisy and contains words that can be non-real vocabulary. We need a clean corpus to have a standard model that can apply to any sub-surface input text. Then we generated a domain-specific corpus related to oil, gas, geology, and sub-surface. This geological corpus gathers mainly from Wikipedia and Schlumberger glossary [1], which we call limited-geological corpus in this dissertation.

Wikipedia: Specific Categories and their subcategories related to the domain of geology, earth science, and petroleum categories were scraped from the English Wikipedia (generally, 2541 categories). These categories contain 14,485 Wikipedia pages, of which 10,079 contain any content. For these pages, the summary and text part of the sections under each page title was scraped. 769 out of 10,079 pages had a non-English title containing an ASCII code inside the URL name, which was discarded. Finally, we have 9310 pages from Wikipedia.

Schlumberger glossary: From SLB Glossary, the content of pages related to 3213 terms was scraped.

1. https://glossary.oilfield.slb.com/

Our standard limited source domain-specific corpus contains 12,523 pages with a total number of 8,627,432 words and 175,856 unique words.

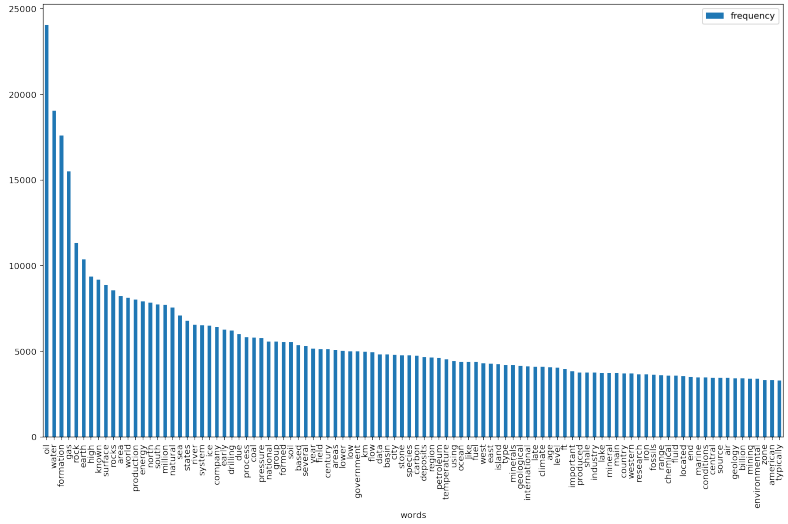


Figure 5.2 – *Top 100 most frequent words from the target domain corpus*

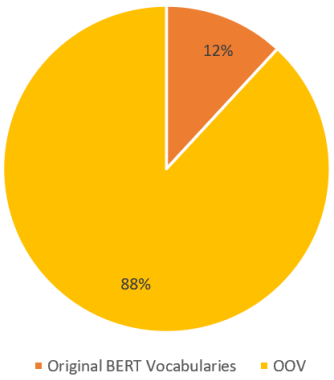


Figure 5.3 – *Domain-specific sub-surface geological words existing in the original BERT model: Common words with original BERT vocabulary (Orange) vs. OOV words from our target domain corpus (Yellow)*

5.4.2 . Vocabulary

As shown in Figure 5.3, the existing vocabulary of our target domain corpus in the original BERT vocabulary is just 11.75%. It means that most of the words in our domain are OOV, even though we have a very limited sources corpus. There is no pre-trained embedding for about 88% of vocabularies, and generating a vocabulary from sub-word pieces can lead to less meaningful content, especially when the number of OOV is high. To the original BERT vocabulary, a domain extension

vocabulary is added. First, using WordPiece [126] and the target domain corpus, an extension vocabulary is extracted. at the same time, the original generic vocabulary used by the original BERT remain unchanged. Any frequent vocabulary of our corpus that already existed in the original BERT vocabulary was eliminated to ensure no duplication in the extension vocabulary exists. Then, the extended vocabulary is included as a corresponding embedding layer, which can be improved during pre-training and is first initialized randomly. For tokenizing input text, a total of 43931 tokens from the vocabulary are used: 30,522 (original) and 13,409 (extension) tokens.

5.5 . Pre-training GeoBERT

The BERT model pre-training is done through two tasks: 1) Masking, and 2) Next Sequence Classification. In masking, a predetermined number of the input sequence's words are removed, changed to a random word, or left intact. The original words for these terms are then predicted by the model. Note that only the modified words need to be predicted by the model, not the complete denoised text. The model learns a representation for each word in the sequence because it is still determining which words it will be questioned about. The process of next sequence classification, also known as next sentence prediction, involves sampling two sequences of roughly 256 words that either directly (a) follow each other in the corpus or (b) are both selected at random and don't follow each other. The model should predict which case happened: a or b.

5.5.1 . Data set and Setup

We must extract a training dataset from our corpus with the same description that we provided above in order to be able to provide the corpus as the input to the model. We consider a maximum of 128 tokens as the sequence length. First, we need to have our own sentence recognizer, which creates a flag for those sentences that follow each other's and are from the same documents. Then we need to check the size of how many of the sentences that follow each other are at most 128 tokens. We create a sequence of those eligible sentences as part of our dataset for part (a). Then we randomly mix sentences from dataset (a) that do not follow each other, and their lengths are smaller than 128. Now we have our generated (a) and (b) training datasets. It contains 150000 sequences with the size of about 128 tokens. We feed these sentences to pre-train the GeoBERT model on top of the embedded pre-trained BERT model.

5.5.2 . Experiment and Analysis

The instance of original BERT in this section refers to Bert-base-uncased. In GeoBERT, the 'extension module' comes from the exBERT model [124], which has the same transformer-based architecture as BERT [100] with smaller sizes. Liu et al. [130] show that carefully designed pre-training choices like dynamic masking,

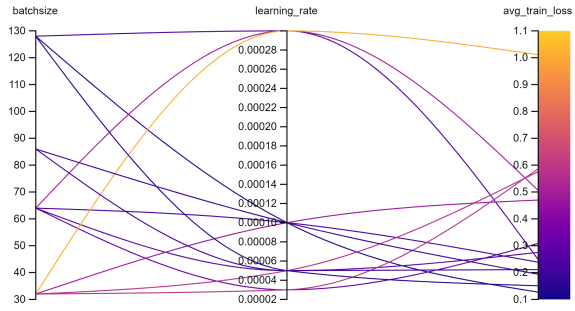


Figure 5.4 – *Parameter importance with respect to average train loss*

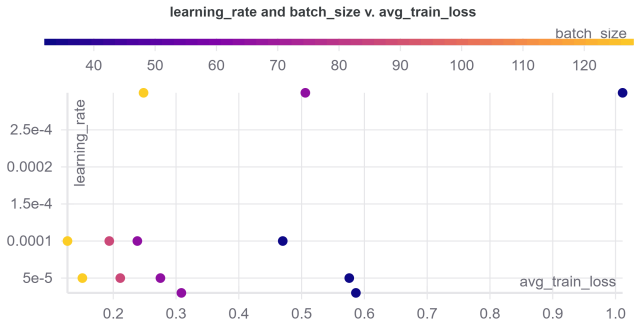


Figure 5.5 – *Parameter importance with respect to average train loss*

large batch sizes, more pre-training steps, and long input sequences enhance the model's performance. The term "batch size" refers to the total number of training instances in a batch. As stated in the exBERT model [124], just the extension module and the weighting block are altered during pre-training. The copy of the original BERT's component in the architecture stays fully unchanged. We have tried different batch sizes and learning rates, as you can see in Figures 5.4 and 5.5. The best batch size and learning rate for pre-training are 128 and $1e$-4, respectively. The experiments are done on 4 V100 NVIDIA GPUs. The training uses the Adam optimizer ($\beta 1 = 0.9$, and $\beta 2 = 0.999$) as it is used in exBERT model. Input length and batch size are both set to 128. It means the length of sentences cannot exceed 126 tokens (2 tokens are reserved for [CLS] and [SEP], as it is shown in figure 5.1). The constructed domain-specific corpus (limited-geological corpus), as mentioned in section 5.4.1 is used for the pre-training the GeoBERT model. The batch size is an essential parameter concerning having minimum loss in training the BERT model. As shown in Figure 5.4, The larger the batch size, the lower the loss. Our model learns best with the learning rate of $1e$-4. The loss changes based on the value of different parameters are shown in Figures 5.4 and 5.5. We could not try the bigger batch size, such as 256 since it needs bigger computation power (GPUs).

Figure 5.6 shows the GPU performance on training GeoBERT with different values of parameters. As it is expected, the smaller batch size causes longer training

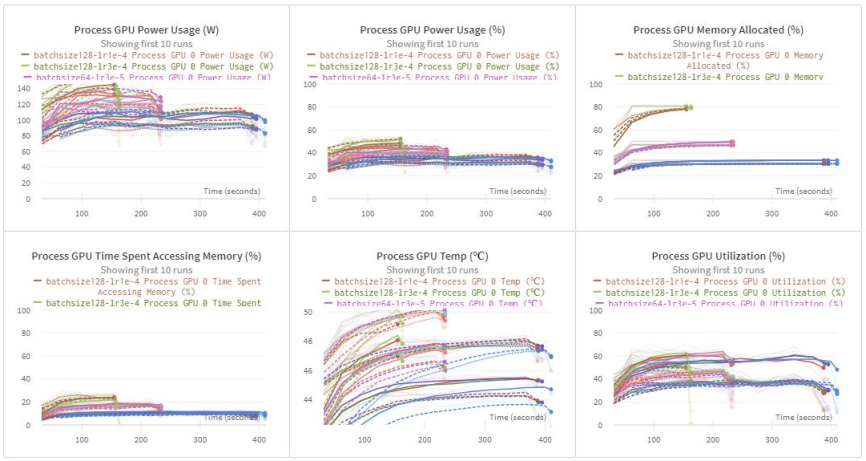


Figure 5.6 – *GPU performance in training GeoBERT with different parameters*

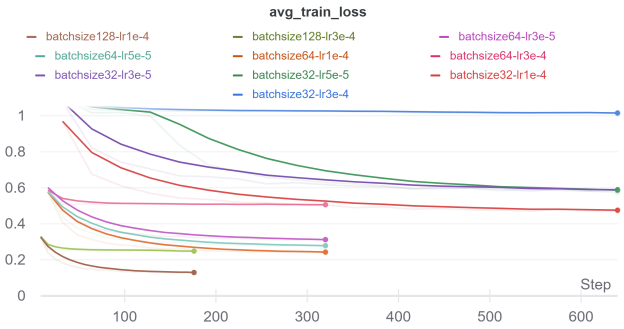


Figure 5.7 – *Average train loss per epoch*

time.

The average training loss per epoch and per batch are shown in Figures 5.7 and 5.8.

Now that we have pre-trained the model let's see how embeddings change in the GeoBERT vs. BERT model in the following section (Section 5.5.2).

Pre-trained words embedding in GeoBERT vs. original BERT model

To better understand the embeddings, we cluster pre-trained words embedding in original BERT vs. GeoBERT. Figure 5.9 shows the T-SNE visualization of clusters of embedding in the last layer of the BERT model based on cosine similarity. If we zoom in on the middle part of Figure 5.9, we can see some clusters of words that have similar meanings in better visualization (Figure 5.10):

Figure 5.11 shows the T-SNE visualization of embedding clusters in the last layer of the GeoBERT model for the same top 4000 most frequent words of the targeted corpus.

Figure 5.8 – *Average train loss per batch*

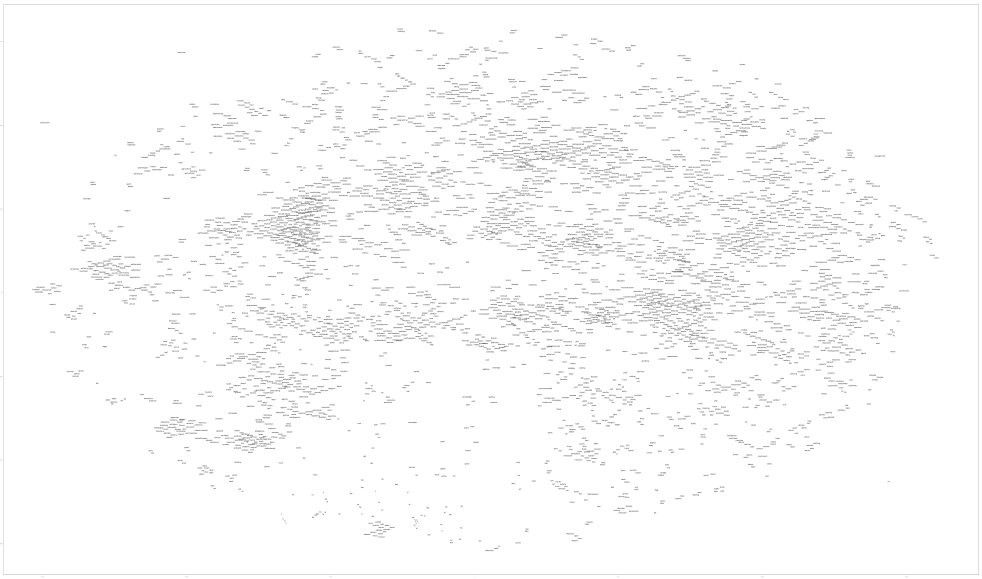


Figure 5.9 – *Clusters of top 4000 most frequent words embeddings in our target corpus which are not OOV in original BERT model*

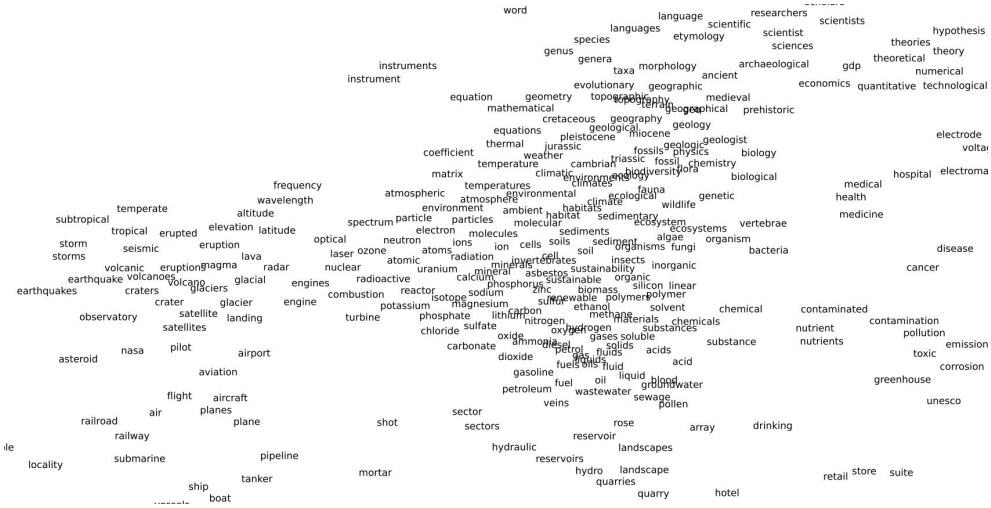


Figure 5.10 – *A closer look to some clusters of word embedding in original BERT*

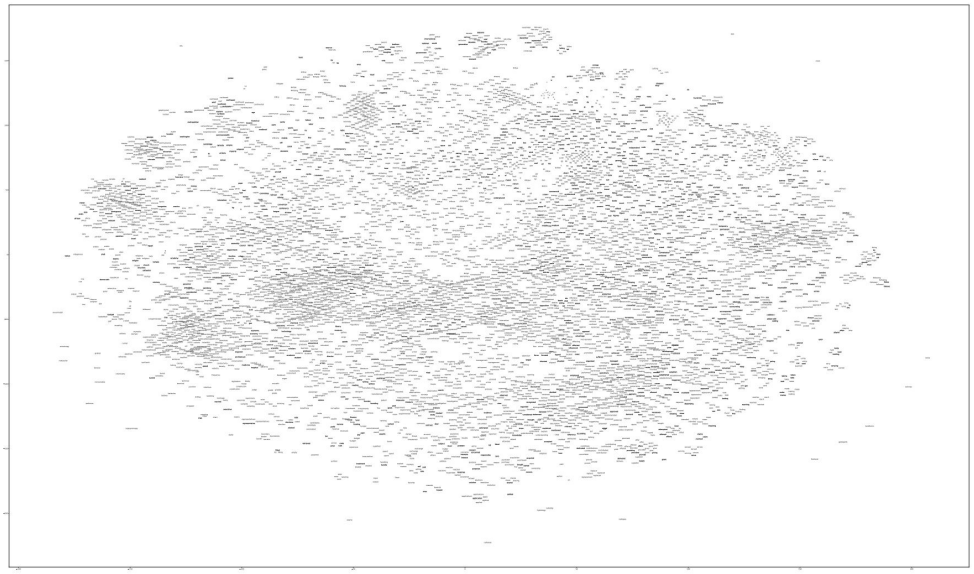


Figure 5.11 – *Clusters of top 4000 most frequent words embeddings in our target corpus which are not OOV in GeoBERT model*

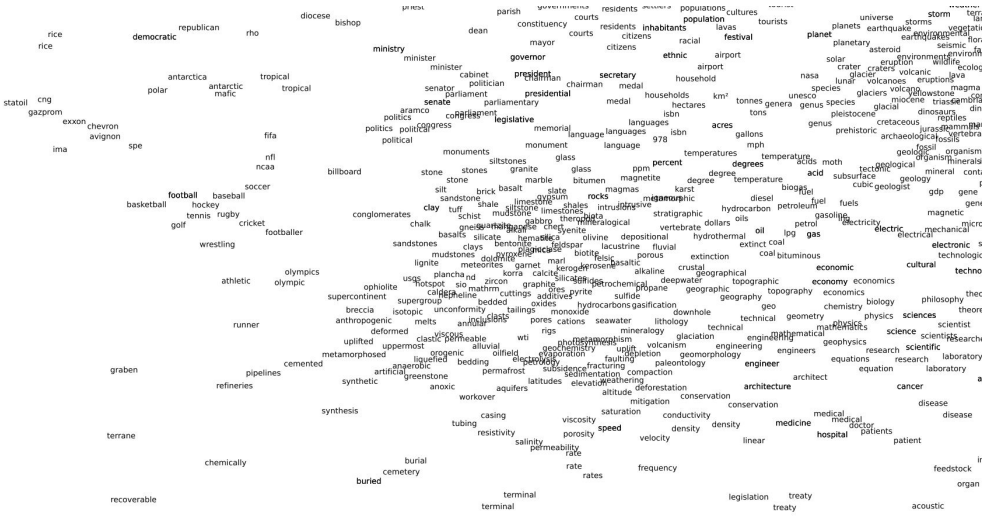


Figure 5.12 – *A closer look to some clusters of word embedding in GeoBERT*

Figure 5.12 shows the T-SNE visualization of embedding clusters in the last layer of the GeoBERT model based on cosine similarity.

Let's look at some specific words shared in both models' embedding. Figure 5.13 shows the part of embeddings cosine distance in BERT model.

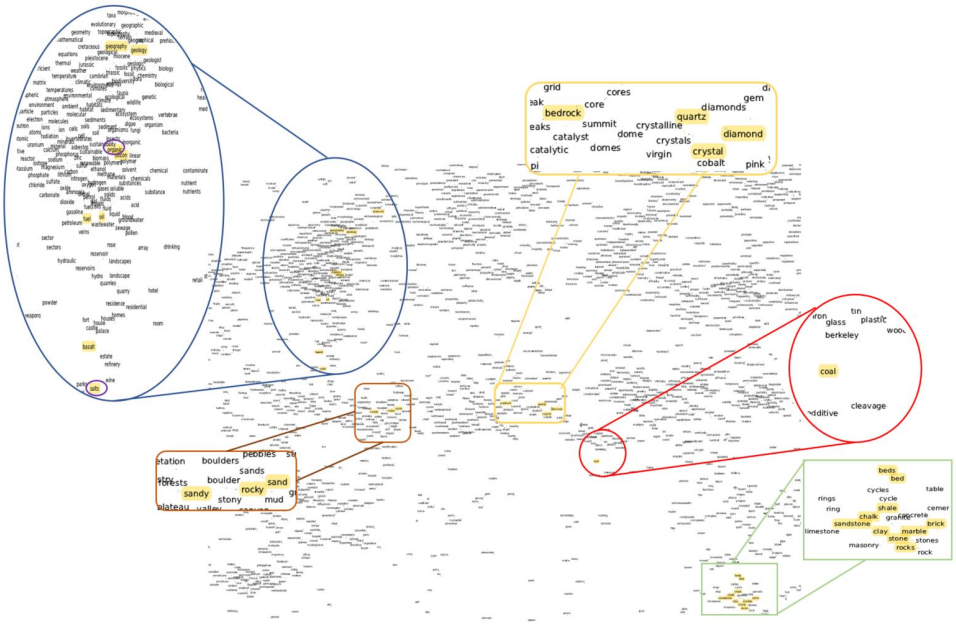


Figure 5.13 – *A closer look at the cosine distance of some specific words in BERT*

As shown, *"coal"* is far from *"oil"* and the cluster of words that contains the words such as *"rocks"*, *"clay"*, *"chalk"*, and *"shale"* are far from the cluster containing words such as *"oil"* and *"fuel"*. The same colors identifying the closest

embedding vectors for a group of words and the same color in two different model shows how the distance among the same group of words changes.

Table 5.1 – *Nearest points to the word Oil in original space of embedding in order*

Selected specific words				All vocabularies			
BERT		GeoBERT		BERT[a]		GeoBERT[a0]	
Tokens	Distance[b]	Tokens	Distance	Tokens	Distance	Tokens	Distance
oils	0.574	oils	0.572	oils	0.557	oils	0.541
petroleum	0.622	petroleum	0.624	water	0.564	petroleum	0.591
water	0.677	water	0.685	petroleum	0.575	water	0.663
gas	0.688	gas	0.694	gas	0.604	gas	0.667
energy	0.747	coal	0.752	black	0.635	energy	0.726
coal	0.750	energy	0.783	energy	0.635	coal	0.727
fuel	0.760	fuel	0.792	power	0.639	fuel	0.730
air	0.766	air	0.792	air	0.640	grease	0.739
powder	0.768	hydrocarbon	0.793	food	0.653	air	0.746
black	0.775	power	0.802	white	0.658	power	0.750
food	0.779	lgp	0.804	a	0.659	sugar	0.757

[a]all vocabularies [b]Cosine distance

However, as you can see in Figure 5.14, which is the showing the cosine distance of the exact words in the GeoBERT model, the word *"coal"* is becoming so close to the word *"oil"*, as well as the cluster of words containing *"rocks"*, *"clay"*, *"chalk"* and *"shale"*.

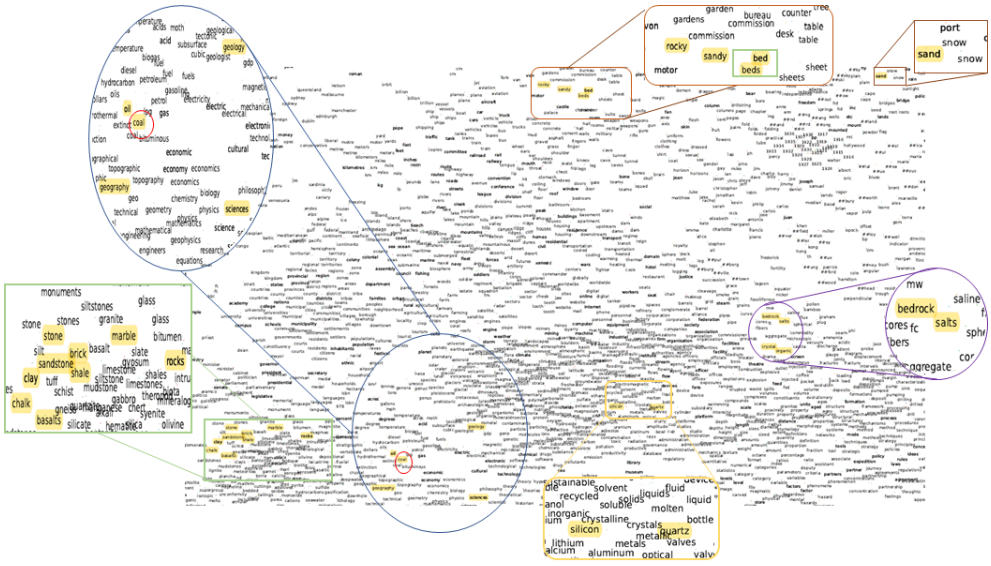
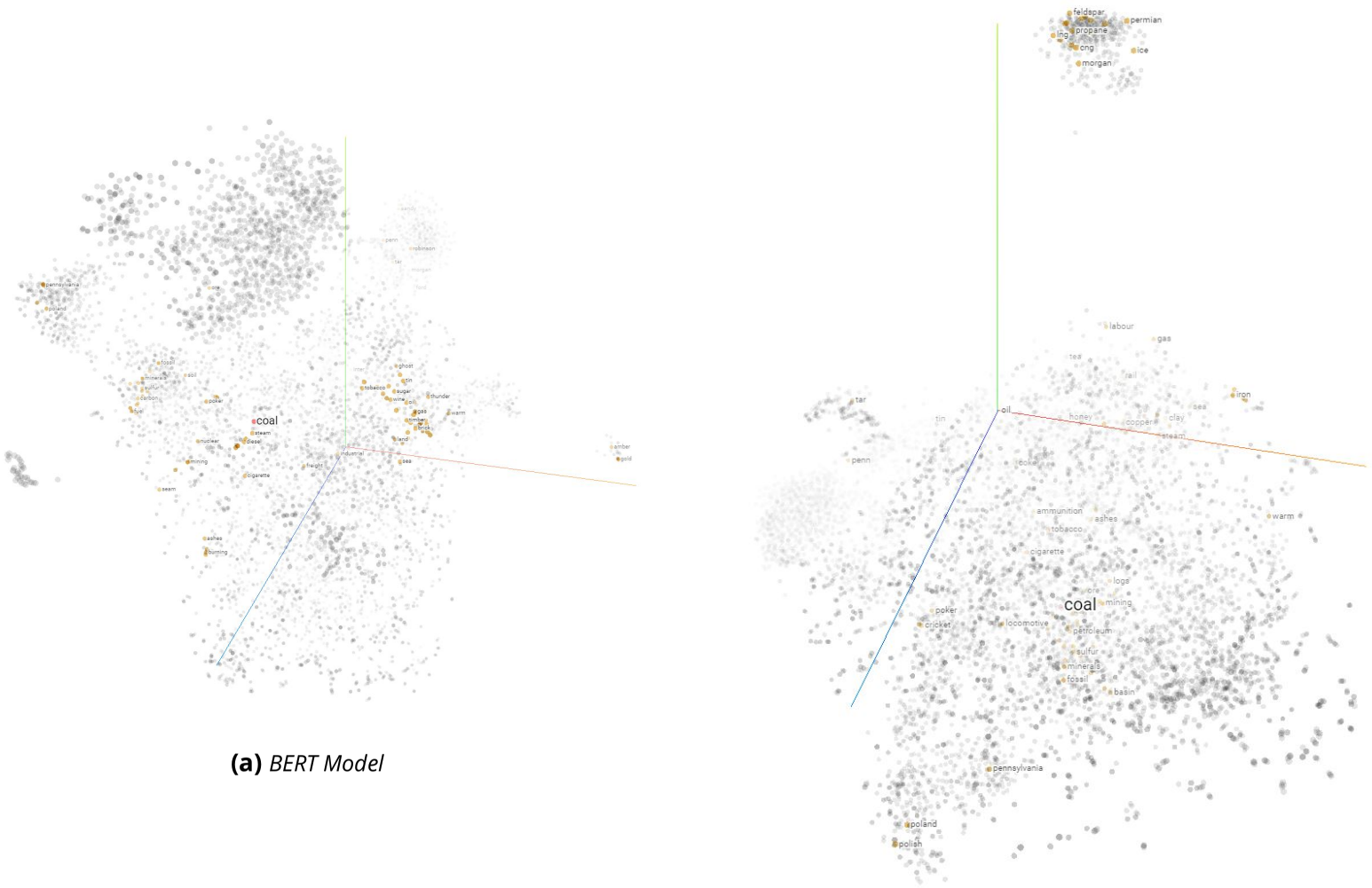


Figure 5.14 – *A closer look at the cosine distance of some specific words in GeoBERT*

The cosine distance changes for some specific words before and after adding extended words in the GeoBERT vs. original BERT model are shown in Figures 5.15a and 5.15b.



(a) *BERT Model*

(b) *GeoBERT Model*

Figure 5.15 – *T-SNE visualization of cosine distance of most similar word embeddings to word Coal*

(a) *BERT Model*

(b) *GeoBERT Model*

Figure 5.16 – *T-SNE visualization of cosine distance of most similar word embeddings to word Shale*

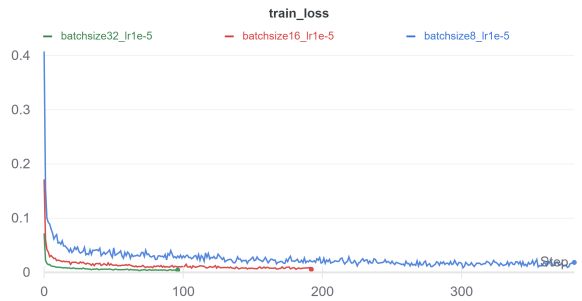


Figure 5.17 – *Loss during 3 epochs for finetuning GeoBERT model*

A better sense of the distance and change of most similar words in two models can be seen in the Tables 5.1 and 5.2.

Table 5.2 – *Nearest points to the word Coal in original space of embedding in order*

BERT		GeoBERT		BERT[a]		GeoBERT[b]	
Tokens	Distance[c]	Tokens	Distance	Tokens	Distance	Tokens	Distance
oil	0.747	oil	0.752	oil	0.667	sandstone	0.638
mining	0.757	mining	0.761	mining	0.688	basalt	0.650
copper	0.766	copper	0.770	copper	0.697	limestone	0.666
fuel	0.780	petroleum	0.783	gas	0.703	sedimentary	0.677
petroleum	0.783	fuel	0.783	colliery	0.711	cambrian	0.696
gas	0.783	gas	0.783	fuel	0.711	geologist	0.706
steam	0.789	steam	0.792	iron	0.713	joshi	0.708
tobacco	0.791	tobacco	0.792	gold	0.716	miocene	0.710
diesel	0.791	diesel	0.793	land	0.717	refinery	0.713
miners	0.802	miners	0.802	steel	0.721	westphalia	0.713
freight	0.802	freight	0.804	wood	0.729	groundwater	0.716

[a]all vocabularies; [b]all vocabularies; [c]Cosine distance

5.6 . Finetuning GeoBERT

After fine-tuning various pre-trained models for the NER problem, we compare their performance. The best configuration for this downstream task is fine-tuning all layers with a learning rate of $1e$-5 and batch size of 16 for three epochs on a dataset provided from a part of sub-surface reports mentioned in section 5.6.1.

This learning rate and batch size value are chosen based on tuning hyper-parameters for the fine-tuning model for the NER task, as seen in Figure 5.17.

5.6.1 . Data set and Setup

The generated annotated dataset in the previous chapter (Chapter 4) is very noisy, and the annotated labels are also noisy. One way is to see the result of the NER task on our annotated dataset. However, another dataset was labeled during a parallel study in Schlumberger [131]. In this study, we used a part of the dense part of documents presented in OCRANA format (about 1000 report documents). The same methodology described in section 4.5is used to annotate the

Table 5.3 – *Geological entity types and their description*

Entity Name	Entity Description
Period	one of the divisions of geologic time that allows for rock cross-referencing.
Epoch	a period of time in geology during which a group of rocks is formed.
Age	one of the geologic divisions that splits an epoch into smaller chunks.
Formation	a body of rock with a consistent set of physical properties.
Depth_Interval	boundaries of a lithostratigraphic unit which indicates the lithologic change.
Interval	boundaries of lithostratigraphic unit exactly like Depth_Interval without measurement unit.
WELL_ID	an identifier for wellbores.

Table 5.4 – *Data sets for Fine-tuning on NER task.*

Entity	**Noisy set**	**Clean set**	**Eval set**	**Test set**
AGE	11243	130	118	280
EPOCH	19366	166	156	360
FORMATION	18424	159	167	381
PERIOD	7416	79	87	166
WELL_ID	15754	125	151	345
INTERVAL	9218	93	83	189
DEPTH_INTERVAL	4258	40	56	92
TOTAL	**85679**	**792**	**818**	**1813**

texts. The matcher component finds the corresponding chunks where the dictionary or RegEx matches the specific input. The list of entities selected as our entities is shown in Table 5.3. The entities identified with blue are the common entities with those that were annotated in Chapter 4. However, as it can be seen, the main difference is about the entities with numerical values or special characters such as WELL_ID, Depth_Interval, or Interval. For example, 30/2a-81 is a typical well identifier, or a depth interval indicates a boundary of numerical value as depth measurement: -6990 to -7013 ft TVDSS or Interval which is the same as without unit of measurement: -6990 to -7013. As was explained in Chapter 4, these kinds of entities cannot be generated with the GAGNER approach.

The annotated dataset used BIO format. After sentence recognition, more than 125,000 sentences with approximately 227,000 entities were annotated. However, to generate the training/evaluation/test dataset, the entire data was not selected. Sentences were selected randomly to create these data sets. Noisy training, clean training, and test set contain 50000, 500, and 1000 random sentences, respectively. These sentences contain approximately 86,000 annotated entities (Table 5.3). These annotated datasets were used to fine-tune the NER task for the BERT and GeoBERT model5.4.

The code of the experiment is implemented in Python 3.6.7. We used four vCPUs with 15 GB RAM deployed on the Google Cloud Platform (PyTorch version 1.9). To recognize the sentences from the text of the reports, we used the Spacy package [107].

Table 5.5 – *finetuning BERT vs. GeoBERT Model on NER task*

BIO format of entity types	BERT Model			GeoBERT Model		
	P	R	F1	P	R	F1
B-AGE	0.97	1.00	0.98	0.99	1.00	1.00
B-DEPTH_INTERVAL	0.61	0.91	0.73	1.00	0.95	0.97
B-EPOCH	0.92	0.93	0.92	0.95	0.93	0.94
B-FORMATION	0.85	0.84	0.85	0.88	0.85	0.87
B-INTERVAL	0.61	0.84	0.70	0.72	0.89	0.80
B-PERIOD	0.97	0.87	0.92	1.00	0.89	0.94
B-WELL_ID	0.72	0.98	0.83	0.81	0.96	0.87
I-DEPTH_INTERVAL	0.92	0.86	0.89	1.00	0.93	0.96
I-EPOCH	0.55	0.40	0.46	1.00	0.17	0.29
I-FORMATION	0.85	0.80	0.82	0.94	0.82	0.87
I-INTERVAL	0.73	0.64	0.68	0.80	0.80	0.80
I-PERIOD	0.00	0.00	0.00	0.00	0.00	0.00
I-WELL_ID	0.94	0.18	0.31	0.98	0.36	0.53
O	0.98	0.98	0.98	0.98	0.99	0.99
Micro Average	**0.85**	**0.86**	**0.85**	**0.96**	**0.96**	**0.96**
Macro Average	**0.76**	**0.73**	**0.72**	**0.86**	**0.75**	**0.77**

5.6.2 . Experiments and Analysis

We fine-tune BERT and GeoBERT on the same dataset 5.6.1, and you can see the difference in the average losses in the two models in Figure 5.4. As mentioned, we chose the best hyperparameters for our GeoBERT model, which means a learning rate of $1e$-5 and batch size of 16. When we talk about the GeoBERT model, we mean the model with these parameters. The average training loss during fine-tuning both models of GeoBERT and BERT per batches and epochs is shown in Figure 5.18. Also, the accuracy and F1-score for both models can be seen in Figures 5.19 and 5.20, respectively.

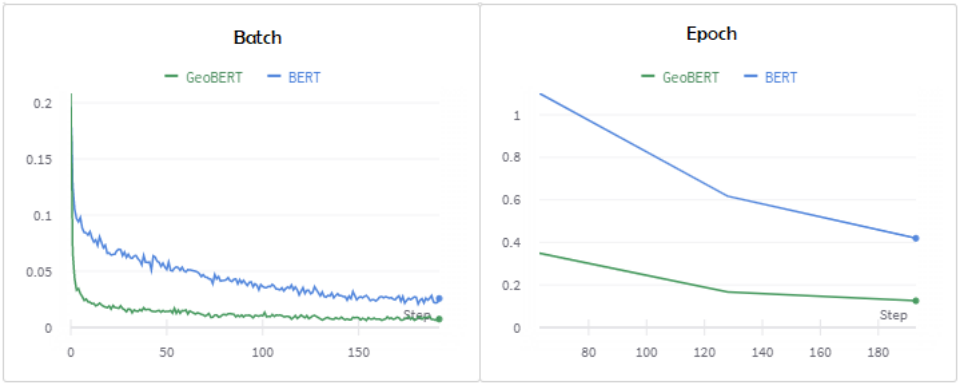


Figure 5.18 – *Average of loss during finetuning BERT vs. GeoBERT per batches and epoches for NER task*

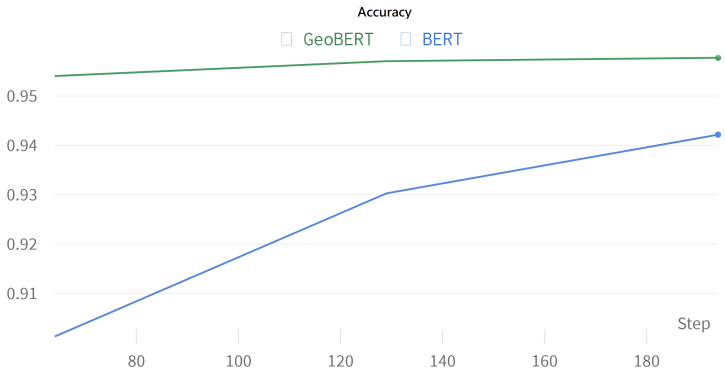


Figure 5.19 – *Accuracy of BERT vs. GeoBERT Model*

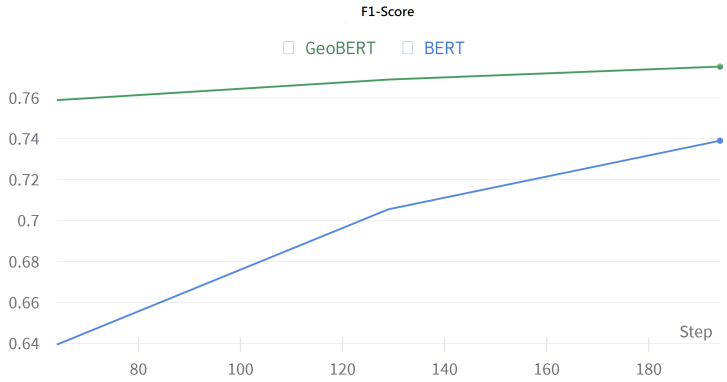


Figure 5.20 – *F1-Score of BERT vs. GeoBERT Model*

Table 5.5 shows the final result of NER for specific entities in BIO format. The GeoBert model performs better on NER tasks regarding precision, recall, and F1-score. In general, the Bert model has a lower performance in recognizing the numerical patterns such as depth_interval and well_id in comparison with named entity types such as age, formation, and . . ., especially in terms of recall. However, even though the same problem happens for the GeoBERT model, we have a much higher recall for the same Bio format entity types in the GeoBERT model.

5.7 . Conclusion

In this chapter, we represented a domain-specific BERT-variant model for the oil and gas industry, specifically for the sub-surface geological domain. We knew pre-training a language model from scratch for a specific domain would cause us to lose the information related to embedding words in the generic BERT model. We need these kinds of generic and common words in a specific domain. Then we decide to adopt a generic BERT model for the oil and gas domain. We present the GeoBERT model by injecting the most frequent domain-specific words as an extended module to the original BERT model. We experimentally show how the extended vocabularies from the oil and gas domain fit in the generic BERT model. Also, it is shown how the new word embeddings for OOV improve the domain-specific word representations, even though the targeted corpus for this adaption is very small since, in general, the sub-surface geological domain is a very limited-source domain. The corpus on which we pre-trained the model is mostly gathered from the geological articles of Wikipedia. For pre-training the generic BERT model, they used the whole of Wikipedia as part of their training corpus. Then, most of the newly injected OOV words to the GeoBERT model may have already been seen in the pre-training of BERT. Still, they never appeared in the vocabulary dictionary of the BERT model in the output. Then somehow, we can conclude that highlighting the most important domain-specific words in the BERT model, even if they have

already been used in the pre-training of the BERT model, can help the specific domains. We apply the GeoBERT model on a NER downstream task, and as a result, we have a higher performance than the generic BERT model. This result indicates that the idea which originally was applied to a large number of biomedical sources can be used for limited source domains as well. In the future, applying this model to other downstream tasks, such as Q&A, can demonstrate that GeoBERT has a better result on different downstream tasks. Also, since the size of GeoBERT is much larger than generic BERT, can we build a distilled version of GeoBERT and see how much the performance on distilled version will drop? Will it be like the performance of the generic BERT model or still be better than the generic BERT model's performance?

6 - Conclusion and Future Work

Throughout this dissertation, we presented different approaches to solving specific problems to understand the sub-surface geological domain given a set of heterogeneous data sources. In This final chapter, we summarize the main achievements resulting from our research; then expose a group of potential improvements together with some future perspectives.

Achievements Sub-surface geological heterogeneous data sources vary in different forms: structured or unstructured sources. The main goal for structured sources is discovering a global schema and the related attributes to that schema to populate it. There are more challenges for unstructured sources. First, we need to find a method for machines to understand the unstructured, heterogeneous format of PDFs and images and extract the related information, primarily domain-specific entities. We propose the PROCLAIM approach to deal with the challenge of structured data. We also propose OCRANA, GAGNER, and GeoBERT as our solutions for turning unstructured data from any documents into a structured format, generating domain-specific gazetteers, and finally, a pre-trained language model for this specific domain, respectively. We can describe the proposed approaches to tackle our challenges and summarize the achievements of each of these proposed approaches respectively as follows:

- We present PROCLAIM (PROfile-based Cluster-Labeling for AttrIbute Matching), an unsupervised method for matching attributes from a large number of heterogeneous sources in a specific domain. We have shown that our method is efficient on large heterogeneous sources in different domains. As a final output, PROCLAIM can automatically create a set of unique labels assigned to a high percentage of attributes from different heterogeneous sources.
  - Presenting PROCLAIM, a domain-independent method for schema label prediction, performs very well with an increase in the heterogeneity of the datasets. We evaluate our method on two different datasets from two domains, and the results are promising.
  - Defining the concept of attribute profile by taking into account the data type using: (i) the statistical distribution and the dimension of the attribute's values and (ii) the name and textual descriptions of the attribute. These properties give a unified representation to each attribute.
  - Presenting the Extended OPTICS, an extended version of OPTICS, by introducing a dynamic minimum number of points in place of a static one for clustering the attributes profiles.

  - Illustrating the concept of labeling function to name the clusters of similar attributes from different sources. The labeling function considers (i) the attribute descriptions and (ii) the attribute names for each cluster. Then it automatically assigns a label to each cluster as the final name of the similar attributes in the global schema of the domain.
  - By using PROCLAIM, about 86% of attributes from 44,000 sources cross-referenced by a name in the generated global schema.
- We propose OCRANA (Optical Character Recognition ANAlytics), a framework to handle documents provided as images or PDF files processed by different OCR engines to build a unified data model. Generally, OCR tools process these documents and translate them into an intermediate data format which provides the bounding content boxes. OCRANA processes the intermediate data format and gives a unified and more enriched view of the logical and physical layout using the defined data model. This data model is a configurable multi-level element-based scheme annotation to store the documents. To distinguish the dense part from the non-dense part of the texts, we presented the position-based Naïve Bayes model. Our evaluation shows that adding positions to the input of the Naïve Bayes model can enrich the final prediction of the model.
  - Designing a scalable framework that efficiently transforms heterogeneous PDFs or image documents processed by different OCR engines into unified structured information to prepare them for further analysis. This framework relies on a unified data model that allows the representation of different kinds of structures of texts and their visual and content-based properties in a fine-grained element-level scheme.
  - Defining a position-based Naïve Bayes algorithm to leverage the semi-supervised methods to train a layout labeling model to distinguish the dense part of the text from the non-dense part. This position-based Naïve Bayes algorithm uses the probability of the positions of each line and the words and symbols of each group of lines related to each layout label to calculate the probability of the new layout labels given the new lines. Our results show an improvement in comparison with the Naïve Bayes algorithm model.
  - Experimentally showing that OCRANA can facilitate the information extraction of entities and their relations from the texts.
- We propose GAGNER (GAzetteer Generation for Named Entity Recognition) to automatically generate the lists of entities (gazetteers) to create an annotated named entity corpus. We use the vector word representation as the output of shallow neural networks to generate the gazetteers, each representing a specific domain-specific entity type. Researches show that word representation can capture linguistic regularities of both implicit semantic

and syntactic information. In the end, we present a Named entity recognition system made of two modules: (1) Using GAGNER to tag the entities in the corpus to generate a training dataset for neural models; (2) Applying machine learning models on the generated training set. Our result shows highly satisfying results for our named entity recognition system.

  - Presenting GAGNER, a novel unsupervised approach to generating domain-specific gazetteers which do not need external resources such as Wikidata knowledge base to generate the gazetteers; It only uses the corpus text as its input which is a promising approach for low-resource domains.
  - Demonstrating that GAGNER can tag the less-known abbreviations and the wrong written forms (typos) of the words in noisy corpora, making it helpful to annotate a corpus, especially for our generated corpus from the output of OCR engines which does not have very high-quality texts.
  - Illustrating the high quality of generated gazetteers through GAGNER regarding their precision (The precision of generated gazetteers is almost higher than 96%, and by considering lenient precision, almost higher than 98%). These gazetteers can be considered the final named entity type groups or be used to annotate the corpus to train neural models to build a NER system by using the minimum resource (only using the corpus text).
  - BERT model by using the training set annotated by these generated gazetteers after fine-tuning for the NER task has an increase of 2%, 8%, and 4.5% for precision, recall, and F-score, respectively, in comparison with the CRF method, which is one of the state-of-the-art models for NER.

- We present GeoBERT to adapt the generic BERT model to a specific domain with limited sources. We notice that a limited sources domain, such as the sub-surface geology domain, can benefit from an extension of the pre-trained model with new vocabularies. New additive vocabulary (out-of-vocabulary words (OOV)) from domain-specific sources are added to an extension module to adapt an integrated embedding for the new domain in the context of the initial BERT embeddings of general vocabularies.
  - Pre-training GeoBERT as a domain-specific BERT-variant language model for the sub-surface geological domain. GeoBERT is adapting the generic BERT model to this specific domain containing a limited number of sources. We see that the generic BERT model can benefit from integrating an extension module to solve the issue of unseen vocabulary (OOV) of the specific limited-source sub-surface domain. The final result for the same NER downstream task compared with the generic BERT model shows improvements.

— The size of vocabulary in GeoBERT has increased by 44% compared to the original vocabulary of the BERT model. The newly added vocabularies are all domain-specific and related to sub-surface geology (13,409 new vocabularies).

— The precision, recall, and F-score of geological entity recognition through the GeoBERT model has increased by 13%, 3%, and 7%, respectively, compared with the original BERT model.

Future Works The subject for this dissertation is very wide. Any problem we tackled in this dissertation could have been a separate research question with a deeper concentration. Even though all the challenges are important subjects for understanding the sub-surface geological domain, each step has many improvements. Many other new steps also can be taken to have a more complete and holistic framework to understand domain-specific heterogeneous sources such as the sub-surface geological domain. Some of these improvements which can interest us are as follows:

— We defined the concept of attribute profile by taking into account the data type using two important types of information: (i) the statistical distribution and the dimension of the attribute's values, and (ii) the name and textual descriptions of the attribute. These properties give a unified representation to each attribute. However, these properties are unavailable for all of the variables in our data sources regarding missing values or lack of provided metadata. Then we can think of a profiling method for those variables depending on the existing information.

— The labeling function is considered based on: (i) the attribute descriptions and (ii) the attribute names for each cluster to automatically assign a label to each one as the final name of the similar attributes in the global schema of the domain. This depends on bigram and trigram terms generated from a domain-specific corpus. This corpus can be enriched, which means more bigram and trigram terms that can be used for labeling the clusters. Also, we could make a connection between the GeoBERT model and the PROCLAIM approach, which is definitely interesting and a very genuine idea that can be explored in the future.

— The final output of OCR engines contains lots of misspellings and errors when extracting the texts from image versions of documents. In the OCRANA framework, we keep the confidence rate of the OCR engine for each character. A simple way to clean the extracted text as much as possible is by considering a threshold for the confidence rate. When the confidence rate is lower than this threshold, the alternative character for this character can be proposed to have a correct form of original words. We can have an alternative list of alternative vocabularies, which, based on the context, have a higher

probability of being the alternative words. Applying Bayesian methods can help to have the best alternative. In this case, we will have much cleaner output from OCR and high-quality text in our OCRANA model.

— Another way to do text cleaning in the OCRANA framework is to apply the GAGNER approach. GAGNER will find the best alternative for words containing at least one character with a low confidence rate. In GAGNER, we already have the similarity of available alternatives, which can help to clean the text.

— To generate a weakly labeled dataset of the structure of documents (a dataset of binary layout labels), we proposed a position-based Naïve Bayes algorithm, which considers the position as one important feature to label the lines. Even though this method is promising, it can be very interesting to see if we can inject the positions as new OOV words into a generic pre-trained language model. Exactly like how we pre-train the GeoBERT model, but this time by adding the categories of positions as vocabularies in extended vocabularies. Also, training a fine-grained multi-labeling model for document structure recognition is a very important task that should be done in the future.

— GAGNER has the capacity to be applied to any language without any constraints. This approach needs to generate static word representation models in any specific language and generate the gazetteers. We have shown how this model is useful for a limited source domain like a geological subsurface. It could be interesting to explore and prove this capacity with more experiments in other domains and different languages.

— The size of the final GeoBERT model is huge because of the number of extended vocabularies and weights of the extension module. One way to have a smaller size for this model is to use a base model with a lower number of parameters, such as the distilled version of BERT, distillBERT (the original BERT has 110 million parameters, and distillBERT has 66 million parameters).

— Another way to have a smaller GeoBERT model is to distill the GeoBERT model directly. A smaller GeoBERT model is very interesting because a distilled model uses less space and operates more quickly.

7 - A summary of the thesis in French

Ce mémoire décrit un problème dans l'industrie pétrolière et gazière où le processus de forage de nouveaux puits productifs est coûteux, long et destructeur pour l'environnement. Les compagnies pétrolières et gazières recueillent et étudient des données sur le sous-sol avant et pendant le forage afin de forer un ensemble de puits rentable. Ces données sont stockées dans différents formats, notamment des PDF, des images, des tableaux et des bases de données structurées. Le texte indique que de nombreux types de recherche se concentrent sur le développement de méthodes de gestion, d'exploitation et d'utilisation efficaces de ces données non structurées. Cette thèse vise à étudier les approches permettant de mieux comprendre le domaine géologique, de rassembler et de fusionner des sources de données structurées hétérogènes, de reconnaître la structure de sources non structurées hétérogènes et d'effectuer la reconnaissance d'entités spécifiques à un domaine afin d'accélérer le processus de recherche d'informations. Cette thèse comprend quatre chapitres principaux, ainsi qu'une introduction et une conclusion. Cette thèse a été menée sur un jeu de données de 44 000 sources de données hétérogènes structurées et 3 400 sources de données hétérogènes non structurées, ce qui a donné lieu à des contributions organisées par chapitre. Le chapitre deux présente PROCLAIM, une méthode non supervisée de mise en correspondance d'attributs provenant d'un grand nombre de sources hétérogènes dans un domaine spécifique. La technique utilise des profils d'attributs et l'étiquetage de grappes pour attribuer des étiquettes à des attributs similaires provenant de différents schémas, représentant l'essence de chaque groupe et définissant le schéma global. Le chapitre trois présente OCRANA, un cadre évolutif permettant de transformer efficacement des documents PDF ou des images hétérogènes en informations structurées unifiées à l'aide d'un modèle de données représentant les résultats intermédiaires et finaux et d'un algorithme de classification Naive Bayes basé sur la position. Le chapitre quatre propose un système de reconnaissance d'entités nommées basé sur la génération automatique de répertoires de noms, appelé GAGNER. Le système utilise des méthodes de réseaux neuronaux peu profonds pour générer automatiquement des répertoires toponymiques et un modèle BERT pré-entraîné pour entraîner un modèle de reconnaissance d'entités nommées personnalisé. Le chapitre 5 présente GeoBERT, un modèle linguistique BERT-variante spécifique à un domaine pour l'industrie pétrolière et gazière. Il vise à combler le manque d'un modèle BERT générique dans un nouveau domaine en l'affinant sur un grand corpus de textes liés au pétrole et au gaz. La thèse se termine par une discussion sur les conclusions et les travaux futurs au chapitre 6. La première partie de ce mémoire présente une étude de recherche sur l'appariement des schémas, le processus de création d'une vue globale de divers schémas développés indépendamment. Cette partie aborde les principaux problèmes de l'intégration des données, notamment l'hétérogénéité des données,

les ensembles de données réelles bruyantes et la nécessité d'un schéma global. Une solution proposée, PROCLAIM (PROfile-based Cluster-Labeling for AttrIbute Matching), est une méthode non supervisée de mise en correspondance d'attributs provenant de nombreuses sources hétérogènes dans un domaine spécifique. La méthode utilise des profils d'attributs et le regroupement OPTICS étendu pour créer automatiquement un ensemble d'étiquettes uniques attribuées à un pourcentage élevé d'attributs et une méthode indépendante du domaine pour la prédiction des étiquettes de schéma. L'étude démontre l'efficacité de PROCLAIM avec deux ensembles de données différents provenant de deux domaines différents et valide la qualité des étiquettes générées avec des experts du domaine. PROCLAIM implique la définition d'un concept de profil d'attribut et l'extension de l'algorithme de clustering OPTICS. Le profil d'attribut prend en compte le type de données, la distribution statistique, les descriptions textuelles et les noms d'attributs. Cette approche propose une méthode indépendante du domaine pour la prédiction des étiquettes de schéma et utilise des experts du domaine pour valider les étiquettes générées avec des résultats prometteurs. PROCLAIM est une méthode efficace de mise en correspondance des schémas qui fournit un schéma d'attributs cohérent et spécifique à un domaine. Elle a le potentiel de rassembler automatiquement plus de 80

La technologie OCR (reconnaissance optique de caractères) a été largement utilisée pour traiter et extraire des informations des PDF et d'autres formes de documents. Le processus consiste à convertir ces documents dans un format lisible par une machine, tout en préservant la structure initiale du document. De nombreux systèmes OCR ont été développés ces dernières années pour la lecture automatique de textes. Cependant, l'OCR pose encore des problèmes, comme le fait de ne fournir que les contours des caractères, des taux de confiance faibles pour des caractères spécifiques en raison du texte barré ou de la mauvaise qualité des images numérisées, et la nécessité de disposer d'ensembles de données à grande échelle dans un format compréhensible par la machine. L'objectif est d'extraire le texte des images tout en préservant la structure du document, de stocker les informations dans un schéma hiérarchique et de traiter la faible qualité des résultats de l'OCR. Le cadre proposé par OCRANA vise à relever les défis posés par la structure hétérogène des documents afin de fournir une structure de document à grain fin au niveau des éléments. OCRANA utilise un moteur de reconnaissance optique de caractères (OCR) pour traiter les documents sous forme d'images ou de fichiers PDF et les traduire dans un format de données intermédiaire avec des boîtes de contenu délimitées. OCRANA traite ensuite ce format de données intermédiaire pour produire une vue unifiée et enrichie de l'agencement logique et physique à l'aide d'un système d'annotation configurable basé sur des éléments à plusieurs niveaux. Cette annotation est utilisée pour stocker les documents. En résumé, le cadre OCRANA est conçu pour traiter des documents au format image ou PDF, en utilisant différents moteurs d'OCR pour extraire des informations et construire

un modèle de données unifié. Le modèle de données est configurable, ce qui signifie qu'il peut être étendu pour accueillir des éléments supplémentaires. Il est basé sur trois types de propriétés : 1) des propriétés spécifiques au moteur extraites directement de la sortie du moteur d'OCR, telles que les caractères, les cadres et les numéros de page, 2) des propriétés statistiques calculées sur la base des propriétés spécifiques au moteur, telles que le nombre de mots dans chaque ligne, les cadres de chaque ligne et les espaces entre les mots, et 3) des propriétés indépendantes du moteur, telles que les balises POS, les lemmes et les étiquettes de mise en page, qui peuvent être dérivées de modèles statistiques, linguistiques ou d'apprentissage automatique. Ce modèle de données unifié fournit la base d'une annotation fine au niveau des éléments et permet la reconnaissance automatique de la structure des documents. - L'étiquetage de la mise en page est réalisé à l'aide d'un algorithme de Naive Bayes basé sur la position et des méthodes semi-supervisées, où les fonctions d'étiquetage de la mise en page sont déterminées sur la base d'informations syntaxiques extraites des boîtes de délimitation de chaque ligne. L'algorithme utilise une supervision faible pour attribuer une étiquette de mise en page à chaque ligne sans effort humain. - Le modèle de données utilisé dans OCRANA est configurable et extensible, ce qui signifie qu'il peut être mis à jour avec de nouvelles informations et de nouveaux modèles selon les besoins. - Le cadre d'OCRANA est conçu pour traiter des ensembles de données à grande échelle et peut être utilisé dans une variété d'applications NLP, y compris la reconnaissance de la structure des documents et l'extraction d'informations. - L'utilisation d'une supervision faible dans le processus d'étiquetage de la mise en page réduit la quantité de travail humain nécessaire, ce qui rend le processus d'étiquetage plus efficace et rentable. - Les résultats obtenus avec OCRANA démontrent l'efficacité du cadre proposé dans la reconnaissance de la structure des documents et l'amélioration de la précision des applications NLP. OCRANA comprend également un algorithme Naïve Bayes basé sur la position pour distinguer la partie dense du texte de la partie non dense. Cet algorithme s'appuie sur des méthodes semi-supervisées pour former un modèle d'étiquetage de la mise en page. L'algorithme de Naïve Bayes basé sur la position utilise la probabilité des positions de chaque ligne et des mots et symboles de chaque groupe de lignes liés à chaque étiquette de mise en page pour calculer la probabilité des nouvelles étiquettes de mise en page en fonction des nouvelles lignes. Cette approche améliore le modèle de l'algorithme Naïve Bayes.

OCRANA est conçu pour faciliter l'extraction d'informations sur les entités et leurs relations dans les textes. Le cadre est évolutif, efficace et peut gérer des documents PDF ou des images hétérogènes traités par différents moteurs d'OCR. Les réalisations d'OCRANA ont été démontrées par des évaluations expérimentales qui montrent sa capacité à transformer des données non structurées en informations structurées, permettant une analyse plus poussée et l'extraction d'informations sur les entités et leurs relations à partir des textes.

Le troisième chapitre traite du développement d'un système de reconnaissance

d'entités nommées (NER) dans un domaine à faibles ressources, tel que le domaine de la géologie de sous-sol, où les données annotées et les dictionnaires sont limités. Pour relever ces défis, nous proposons GAGNER. Cette approche non supervisée génère des répertoires géographiques spécifiques à un domaine pour la reconnaissance d'entités en utilisant des représentations vectorielles de mots pour identifier des mots similaires sur la base de leurs informations sémantiques et syntaxiques implicites. GAGNER peut marquer les abréviations et les fautes de frappe moins connues dans des corpus bruyants, et ses résultats peuvent être utilisés pour générer un modèle de NER supervisé utilisant un modèle de transformateurs (BERT). Nous comparons cette approche à un modèle NER traditionnel (CRF) pour démontrer sa performance sur les répertoires géographiques générés. L'approche proposée offre une solution prometteuse pour construire des systèmes de NER dans des domaines à faibles ressources sans dépendre du travail humain et des ressources externes. L'approche GAGNER utilise des techniques statiques de représentation des mots pour créer des vecteurs d'intégration de mots pour un corpus spécifique à un domaine. En effet, ces techniques sont moins gourmandes en ressources que les méthodes d'intégration de mots contextualisés, et elles peuvent donner de bons résultats même sur des corpus plus petits. L'approche GAGNER utilise différentes méthodes, telles que word2vec et fastText, pour générer une liste d'entités nommées liées à différentes classes d'entités et créer des répertoires toponymiques spécifiques à un domaine. Ces répertoires sont utilisés pour annoter le corpus et créer un ensemble de données d'entraînement pour un modèle NER, qui est ensuite entraîné à l'aide d'une représentation contextualisée des mots telle que BERT. Différentes techniques sont disponibles pour générer automatiquement des répertoires toponymiques, y compris des méthodes de regroupement et d'amorçage. GAGNER utilise la similarité entre les vecteurs d'intégration des mots qui contiennent des informations sémantiques et syntaxiques pour les vocabulaires par le biais de modèles neuronaux peu profonds tels que word2vec et fastText. Comme il n'existe pas de référence pour comparer le pourcentage d'entités nommées existantes liées à un répertoire toponymique dans le corpus généré par GAGNER, l'exactitude des répertoires toponymiques générés est évaluée sur la base de la précision, qui est calculée en évaluant manuellement chaque répertoire toponymique avec l'aide de géologues. Les résultats de précision d'au moins 96À l'aide de ces répertoires toponymiques, nous annotons un vaste corpus de NER dans le domaine de la géologie de sous-sol, contenant plus de 663 000 phrases et 915 000 entités nommées dans 7 types d'entités spécifiques au domaine. Enfin, nous avons affiné un modèle BERT en utilisant ce corpus étiqueté et avons obtenu des résultats prometteurs par rapport au modèle NER classique de CRF. Les résultats de l'évaluation montrent une grande précision, ce qui fait des étiquettes annotées un manuel de haute qualité. Cependant, GAGNER est incompatible avec les types d'entités nommées contenant des valeurs numériques ou des symboles suivant un modèle. Nous avons besoin d'une autre approche pour ces types d'entités afin de résoudre cette limitation. Le cha-

pitre cinq présente une nouvelle approche appelée GeoBERT, qui est un modèle de langage BERT-variant spécifique au domaine de la géologie de subsurface. Nous avons adapté le modèle BERT générique à ce domaine spécifique avec des sources limitées en intégrant un module d'extension pour résoudre le vocabulaire non vu (OOV) du domaine de la sub-surface. Si le pré-entraînement des modèles de langage à partir de zéro peut apporter des avantages significatifs dans des domaines spécialisés, il implique également la perte des connaissances intégrées dans les modèles de langage généraux tels que BERT. L'approche GAGNER est utile pour le NER mais présente des limites dans l'extraction des types d'entités nommées contenant des valeurs numériques ou des symboles suivant un modèle. D'autres tâches de TAL, telles que les questions-réponses, peuvent bénéficier de modèles linguistiques pré-entraînés spécifiques à un domaine. En outre, le domaine de la géologie ne dispose pas de dictionnaires, de bases de connaissances ou de corpus complets, et le modèle générique BERT est confronté à des difficultés liées aux mots hors vocabulaire et à l'absence de représentation précise de l'intégration dans des domaines spécifiques. Une option pour améliorer la performance d'un modèle de langage pré-entraîné comme BERT sur un domaine spécifique est d'intégrer des vocabulaires spécifiques au domaine dans le dictionnaire de vocabulaire BERT et de réutiliser les poids du modèle pré-entraîné pour intégrer les enchâssements de vocabulaire du nouveau domaine. Cependant, cette approche peut toujours aboutir à des performances sous-optimales, car le vocabulaire BERT original peut contenir des termes polysémiques ayant des significations différentes dans d'autres domaines. Pour résoudre ce problème, un module d'extension permettant l'interaction entre le modèle générique de l'ORET et les mots spécifiques au domaine doit être utilisé. Ce module peut fournir davantage d'informations contextuelles et améliorer la capacité du modèle à désambiguïser la signification des mots dans le domaine spécifique. Cette approche peut contribuer à réduire les coûts de calcul et les données d'entraînement nécessaires à la construction d'un modèle linguistique spécifique à un domaine à partir de zéro, tout en conservant les connaissances précieuses intégrées dans le modèle pré-entraîné. Ce chapitre démontre que cette approche est compatible avec l'utilisation de petites sources spécifiques à un domaine et qu'elle est plus performante que le modèle générique de BERT, comme le montrent leurs expériences sur le NER. Nous appliquons l'architecture d'exBERT qui est initialement utilisée dans le domaine biomédical. Cette approche n'a jamais été utilisée auparavant dans le domaine de la géologie souterraine, et encore moins avec des ressources limitées. L'architecture de GeoBERT, empruntée à exBERT, comprend le module original du modèle générique BERT et un module d'extension qui ajoute le nouveau vocabulaire du domaine au vocabulaire original du modèle BERT. Le vocabulaire d'extension et la couche d'intégration qui l'accompagne sont pré-entraînés et fusionnés avec les intégrations originales des vocabulaires du modèle original. Le modèle doit être entraîné à nouveau, en tenant compte des mots hors vocabulaire nouvellement injectés. Les deux principales étapes de la

formation d'un modèle spécifique à un domaine basé sur l'architecture exBERT sont l'ajout du module d'intégration d'extension et le processus de préformation. Tout au long de cette thèse, nous avons présenté différentes approches pour résoudre des problèmes spécifiques afin de comprendre le domaine géologique de sous-sol à partir d'un ensemble de sources de données hétérogènes. En résumé, les approches proposées relèvent les défis de l'extraction de données structurées à partir de sources hétérogènes de données géologiques. PROCLAIM fournit une méthode pour faire correspondre les attributs de différentes sources et générer un schéma global pour le domaine. OCRANA est un cadre conçu pour traiter les documents fournis sous forme d'images ou de fichiers PDF traités par différents moteurs OCR et les transformer en informations structurées unifiées. Les moteurs d'OCR traitent les documents et les traduisent dans un format de données intermédiaire qui fournit les boîtes de contenu délimitées. OCRANA traite ce format de données intermédiaires pour fournir une vue unifiée et enrichie de la disposition logique et physique en utilisant une annotation configurable à plusieurs niveaux basée sur les éléments pour stocker les documents. Ensuite, OCRANA, comme GAGNER, peut être utilisé pour extraire des entités spécifiques à un domaine à partir de sources de données non structurées. En outre, GAGNER est une méthode permettant de générer automatiquement des répertoires géographiques, ce qui peut améliorer la précision de la reconnaissance des entités nommées. Et enfin, GeoBERT est un modèle linguistique pré-entraîné qui peut être affiné sur des tâches spécifiques afin d'en améliorer les performances. Les résultats de chaque approche proposée ont été démontrés par des évaluations expérimentales sur des ensembles de données réelles.

References

[1] M. Macintyre, G. Parry, and J. Angelis, *Service design and delivery*. Springer Science & Business Media, 2011.

[2] S. Crittenden, "The lithostratigraphy and biostratigraphy (foraminifera) of the early cretaceous of the southern north sea basin," *PhD Thesis*, 1988.

[3] K. W. Glennie, "Petroleum geology of the north sea: Basic concepts and recent advances," *Blackwell Science*, 2009.

[4] S. Crittenden and S. Crittenden, "Lithostratigraphy and the cretaceous of the north sea a brief outline of concepts it is not easy!," 08 2019.

[5] G. Cope, "What is a well?," *Journal of Petroleum Technology*, vol. 63, no. 02, pp. 18–21, 2011.

[6] Allison and Mandler, "Subsurface data in the oil and gas industry probing beneath the earth's surface for exploration and hazard mitigation," *Petroleum and the Environment, Part 23/24*, 2018. https://www.americangeosciences.org/geoscience-currents/subsurface-data-oil-and-gas-industry.

[7] L. Wu, L. Xue, C. Li, X. Lv, Z. Chen, B. Jiang, M. Guo, and Z. Xie, "A knowledge-driven geospatially enabled framework for geological big data," *ISPRS International Journal of Geo-Information*, vol. 6, no. 6, p. 166, 2017.

[8] Q. Qiu, Z. Xie, L. Wu, and L. Tao, "Gner: A generative model for geological named entity recognition without labeled data using deep learning," *Earth and Space Science*, vol. 6, no. 6, pp. 931–946, 2019.

[9] M. Arman, S. Wlodarczyk, N. Bennacer Seghouani, and F. Bugiotti, "Proclaim: An unsupervised approach to discover domain-specific attribute matchings from heterogeneous sources," in *International Conference on Advanced Information Systems Engineering*, pp. 14–28, Springer, 2020.

[10] H. Zhao, "Matching attributes across overlapping heterogeneous data sources using mutual information," *Journal of Database Management (JDM)*, vol. 21, no. 4, pp. 91–110, 2010.

[11] A. Doan, A. Y. Halevy, and P. M. Domingos, *Learning to map between structured representations of data*. University of Washington, 2002.

[12] A. Gal, "Managing uncertainty in schema matching with top-k schema mappings," in *Journal on Data Semantics VI*, pp. 90–114, Springer, 2006.

[13] E. Taroza, *Schema matching and automatic web data extraction*. Citeseer, 2006.

[14] C. J. Zhang, Z. Zhao, L. Chen, H. V. Jagadish, and C. C. Cao, "Crowdmatcher: crowd-assisted schema matching," in *Proceedings of the 2014 ACM*

SIGMOD International Conference on Management of Data, pp. 721–724, 2014.

[15] A. A. Alwan, A. Nordin, M. Alzeber, and A. Z. Abualkishik, "A survey of schema matching research using database schemas and instances," *IJACSA*, vol. 8, no. 10, 2017.

[16] C. Chen, B. Golshan, A. Y. Halevy, W.-C. Tan, and A. Doan, "Biggorilla: An open-source ecosystem for data preparation and integration.," *IEEE Data Eng. Bull.*, vol. 41, no. 2, pp. 10–22, 2018.

[17] A. Bonifati, G. Mecca, P. Papotti, and Y. Velegrakis, "Discovery and correctness of schema mapping transformations," in *Schema matching and mapping*, pp. 111–147, Springer, 2011.

[18] T. Milo and S. Zohar, "Using schema matching to simplify heterogeneous data translation," in *vldb*, vol. 98, pp. 24–27, Citeseer, 1998.

[19] H.-H. Do, S. Melnik, and E. Rahm, "Comparison of schema matching evaluations," in *Net. ObjectDays: International Conference on Object-Oriented and Internet-Based Technologies, Concepts, and Applications for a Networked World*, pp. 221–237, Springer, 2002.

[20] P. Bohannon, E. Elnahrawy, W. Fan, and M. Flaster, "Putting context into schema matching," in *Proceedings of the 32nd international conference on Very large data bases*, pp. 307–318, Citeseer, 2006.

[21] E. Sutanta, R. Wardoyo, K. Mustofa, and E. Winarko, "Survey: Models and prototypes of schema matching.," *International Journal of Electrical & Computer Engineering (2088-8708)*, vol. 6, no. 3, 2016.

[22] Z. Chen, H. Jia, J. Heflin, and B. D. Davison, "Generating schema labels through dataset content analysis," in *Companion Proceedings of the The Web Conference 2018*, pp. 1515–1522, 2018.

[23] E. Rahm and P. A. Bernstein, "A survey of approaches to automatic schema matching," *the VLDB Journal*, vol. 10, no. 4, pp. 334–350, 2001.

[24] E. Rahm and E. Peukert, "Holistic schema matching.," 2019.

[25] S. Jiang, J. Liang, Y. Xiao, H. Tang, H. Huang, and J. Tan, "Towards the completion of a domain-specific knowledge base with emerging query terms," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pp. 1430–1441, IEEE, 2019.

[26] M. Trabelsi, B. D. Davison, and J. Heflin, "Improved table retrieval using multiple context embeddings for attributes," in *2019 IEEE International Conference on Big Data (Big Data)*, pp. 1238–1244, IEEE, 2019.

[27] M. Trabelsi, J. Cao, and J. Heflin, "Semantic labeling using a deep contextualized language model," *arXiv preprint arXiv:2010.16037*, 2020.

[28] NEXLA, "An introduction to big data formats understanding avro, parquet, and orc," in *NEXLA White paper*, pp. 1–12, 2018.

[29] P. Cerda, G. Varoquaux, and B. Kégl, "Similarity encoding for learning with dirty categorical variables," *Machine Learning*, vol. 107, no. 8-10, pp. 1477–1494, 2018.

[30] D. Rubenstein, W. Yin, and M. D. Frame, *Biofluid mechanics: an introduction to fluid mechanics, macrocirculation, and microcirculation*. Academic Press, 2015.

[31] C. A. Charu and K. R. Chandan, "Data clustering: algorithms and applications," 2013.

[32] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," 1996.

[33] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: ordering points to identify the clustering structure," in *ACM Sigmod record*, vol. 28, pp. 49–60, ACM, 1999.

[34] D. Vohra, "Apache parquet," in *Practical Hadoop Ecosystem*, pp. 325–335, Springer, 2016.

[35] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," *Proceedings of the VLDB Endowment*, vol. 11, no. 3, pp. 269–282, 2017.

[36] P. Varma and C. Ré, "Snuba: automating weak supervision to label training data," *Proceedings of the VLDB Endowment*, vol. 12, no. 3, pp. 223–236, 2018.

[37] T. M. Breuel, "The hocr microformat for ocr workflow and results," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2, pp. 1063–1067, IEEE, 2007.

[38] R. Smith, "An overview of the tesseract ocr engine," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2, pp. 629–633, IEEE, 2007.

[39] T. M. Breuel, "The ocropus open source ocr system," in *Document Recognition and Retrieval XV*, vol. 6815, p. 68150F, International Society for Optics and Photonics, 2008.

[40] C. Clausner, A. Antonacopoulos, and S. Pletschacher, "Efficient and effective ocr engine training," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 23, no. 1, pp. 73–88, 2020.

[41] A. Simon, J.-C. Pret, and A. P. Johnson, "A fast algorithm for bottom-up document layout analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 3, pp. 273–277, 1997.

[42] R. P. Futrelle, M. Shao, C. Cieslik, and A. E. Grimes, "Extraction, layout analysis and classification of diagrams in pdf documents," in *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, pp. 1007–1013, IEEE, 2003.

[43] F. Shafait, D. Keysers, T. M. Breuel, *et al.*, "Layout analysis of urdu document images," in *2006 IEEE International Multitopic Conference*, pp. 293–298, IEEE, 2006.

[44] A. M. Namboodiri and A. K. Jain, "Document structure and layout analysis," in *Digital Document Processing*, pp. 29–48, Springer, 2007.

[45] A. Zulfiqar, A. Ul-Hasan, and F. Shafait, "Logical layout analysis using deep learning," in *2019 Digital Image Computing: Techniques and Applications (DICTA)*, pp. 1–5, IEEE, 2019.

[46] A. J. Ratner, C. M. De Sa, S. Wu, D. Selsam, and C. Ré, "Data programming: Creating large training sets, quickly," *Advances in neural information processing systems*, vol. 29, pp. 3567–3575, 2016.

[47] J. Walker, Y. Fujii, and A. C. Popat, "A web-based ocr service for documents," in *Proceedings of the 13th IAPR International Workshop on Document Analysis Systems (DAS), Vienna, Austria*, vol. 1, 2018.

[48] R. Cattoni, T. Coianiz, S. Messelodi, and C. M. Modena, "Geometric layout analysis techniques for document image understanding: a review," *ITC-irst Technical Report*, vol. 9703, no. 09, 1998.

[49] "Google vision api." https://cloud.google.com/vision/docs/ocr. [Online; accessed 26-September-2022].

[50] A. Mehler, K.-U. Kühnberger, H. Lobin, H. Lüngen, A. Storrer, and A. Witt, *Modeling, learning, and processing of text-technological data structures*, vol. 370. Springer, 2011.

[51] "Parquet." https://developer.ibm.com/hadoop/2016/01/14/5-reasons-to-choose-parquet-for-spark-sql/. [Online; accessed 26-September-2022].

[52] L. Chang, Z. Wang, T. Ma, L. Jian, L. Ma, A. Goldshuv, L. Lonergan, J. Cohen, C. Welton, G. Sherry, *et al.*, "Hawq: a massively parallel processing sql engine in hadoop," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pp. 1223–1234, 2014.

[53] J. Krishnamurthy and T. Mitchell, "Weakly supervised training of semantic parsers," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 754–765, 2012.

[54] S. Helmstetter and H. Paulheim, "Collecting a large scale dataset for classifying fake news tweets using weak supervision," *Future Internet*, vol. 13, no. 5, p. 114, 2021.

[55] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," *The VLDB Journal*, vol. 29, no. 2, pp. 709–730, 2020.

[56] D. Jurafsky and J. H. Martin, "Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition," 2020.

[57] A. Oussous, F.-Z. Benjelloun, A. A. Lahcen, and S. Belfkih, "Big data technologies: A survey," *Journal of King Saud University-Computer and Information Sciences*, vol. 30, no. 4, pp. 431–448, 2018.

[58] S. Salloum, R. Dautov, X. Chen, P. X. Peng, and J. Z. Huang, "Big data analytics on apache spark," *International Journal of Data Science and Analytics*, vol. 1, no. 3-4, pp. 145–164, 2016.

[59] S. Wu, L. Hsiao, X. Cheng, B. Hancock, and C. Re, "Fonduer: Knowledge base construction from richly formatted data," *Proceedings. ACM-SIGMOD International Conference on Management of Data*, vol. 2018, pp. 1301–1316, 2018.

[60] L. Liu, J. Shang, X. Ren, F. Xu, H. Gui, J. Peng, and J. Han, "Empower sequence labeling with task-aware neural language model," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[61] H. Shelar, G. Kaur, N. Heda, and P. Agrawal, "Named entity recognition approaches and their comparison for custom ner model," *Science & Technology Libraries*, vol. 39, no. 3, pp. 324–337, 2020.

[62] A. Passos, V. Kumar, and A. McCallum, "Lexicon infused phrase embeddings for named entity resolution," *arXiv preprint arXiv:1404.5367*, 2014.

[63] H.-J. Song, B.-C. Jo, C.-Y. Park, J.-D. Kim, and Y.-S. Kim, "Comparison of named entity recognition methodologies in biomedical documents," *Biomedical engineering online*, vol. 17, no. 2, pp. 1–14, 2018.

[64] A. P. Quimbaya, A. S. Múnera, R. A. G. Rivera, J. C. D. Rodríguez, O. M. M. Velandia, A. A. G. Peña, and C. Labbé, "Named entity recognition over electronic health records through a combined dictionary-based approach," *Procedia Computer Science*, vol. 100, pp. 55–61, 2016.

[65] C. Zhu, *Machine Reading Comprehension: Algorithms and Practice*. Elsevier, 2021.

[66] J. Shang, L. Liu, X. Ren, X. Gu, T. Ren, and J. Han, "Learning named entity tagger using domain-specific dictionary," *arXiv preprint arXiv:1809.03599*, 2018.

[67] S. Peshterliev, C. Dupuy, and I. Kiss, "Self-attention gazetteer embeddings for named-entity recognition," *arXiv preprint arXiv:2004.04060*, 2020.

[68] D. Vrandečić and M. Krötzsch, "Wikidata: a free collaborative knowledgebase," *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.

[69] T. Rebele, F. Suchanek, J. Hoffart, J. Biega, E. Kuzey, and G. Weikum, "Yago: A multilingual knowledge base from wikipedia, wordnet, and geonames," in *International Semantic Web Conference*, pp. 177–185, Springer, 2016.

[70] T. Mikolov, Q. V. Le, and I. Sutskever, "Exploiting similarities among languages for machine translation," *arXiv preprint arXiv:1309.4168*, 2013.

[71] S. K. Sienčnik, "Adapting word2vec to named entity recognition," in *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, pp. 239–243, 2015.

[72] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," *arXiv preprint arXiv:1603.01360*, 2016.

[73] M. S. Bari, S. Joty, and P. Jwalapuram, "Zero-resource cross-lingual named entity recognition," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 7415–7423, 2020.

[74] Y. Zhang, "Named entity recognition for social media text," 2019.

[75] J. B. Johannessen, K. Hagen, Å. Haaland, A. B. Jónsdottir, A. Nøklestad, D. Kokkinakis, P. Meurer, E. Bick, and D. Haltrup, "Named entity recognition for the mainland scandinavian languages," *Literary and Linguistic Computing*, vol. 20, no. 1, pp. 91–102, 2005.

[76] D. Hanisch, K. Fundel, H.-T. Mevissen, R. Zimmer, and J. Fluck, "Prominer: rule-based protein and gene entity recognition," *BMC bioinformatics*, vol. 6, no. 1, pp. 1–9, 2005.

[77] J. Xie, Z. Yang, G. Neubig, N. A. Smith, and J. Carbonell, "Neural cross-lingual named entity recognition with minimal resources," *arXiv preprint arXiv:1808.09861*, 2018.

[78] E. Pazhouhi, "Automatic product name recognition from short product descriptions," Master's thesis, University of Twente, 2018.

[79] Y. Jiang, C. Hu, T. Xiao, C. Zhang, and J. Zhu, "Improved differentiable architecture search for language modeling and named entity recognition," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3585–3590, 2019.

[80] S. Magnolini, V. Piccioni, V. Balaraman, M. Guerini, and B. Magnini, "How to use gazetteers for entity recognition with neural models," in *Proceedings of the 5th Workshop on Semantic Deep Learning (SemDeep-5)*, pp. 40–49, 2019.

[81] C. H. Song, D. Lawrie, T. Finin, and J. Mayfield, "Improving neural named entity recognition with gazetteers," *arXiv preprint arXiv:2003.03072*, 2020.

[82] S. Rijhwani, S. Zhou, G. Neubig, and J. Carbonell, "Soft gazetteers for low-resource named entity recognition," *arXiv preprint arXiv:2005.01866*, 2020.

[83] M. Collins and Y. Singer, "Unsupervised models for named entity classification," in *1999 Joint SIGDAT conference on empirical methods in natural language processing and very large corpora*, 1999.

[84] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates, "Unsupervised named-entity extraction from the web: An experimental study," *Artificial intelligence*, vol. 165, no. 1, pp. 91–134, 2005.

[85] Z. Zhang and J. Iria, "A novel approach to automatic gazetteer generation using wikipedia," in *Proceedings of the 2009 Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources (People's Web)*, pp. 1–9, 2009.

[86] A. Neelakantan and M. Collins, "Learning dictionaries for named entity recognition using minimal supervision," *arXiv preprint arXiv:1504.06650*, 2015.

[87] M. Joshi, E. Hart, M. Vogel, and J. D. Ruvini, "Distributed word representations improve ner for e-commerce," in *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pp. 160–167, 2015.

[88] U. Naseem, I. Razzak, S. K. Khan, and M. Prasad, "A comprehensive survey on word representation models: From classical to state-of-the-art word representation language models," *Transactions on Asian and Low-Resource Language Information Processing*, vol. 20, no. 5, pp. 1–35, 2021.

[89] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[90] G. Berardi, A. Esuli, and D. Marcheggiani, "Word embeddings go to italy: A comparison of models and training datasets.," in *IIR*, 2015.

[91] A. Sieg, "From pre-trained word embeddings to pre-trained language models — focus on bert," *Medium*, 2019.

[92] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of machine learning research*, vol. 12, no. ARTICLE, pp. 2493–2537, 2011.

[93] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[94] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[95] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[96] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the association for computational linguistics*, vol. 5, pp. 135–146, 2017.

[97] K. Ethayarajh, "How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings," *arXiv preprint arXiv:1909.00512*, 2019.

[98] B. McCann, J. Bradbury, C. Xiong, and R. Socher, "Learned in translation: Contextualized word vectors," *Advances in neural information processing systems*, vol. 30, 2017.

[99] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations." arxiv preprint," *arXiv preprint arXiv:1802.05365*, 2018.

[100] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[101] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," *Computer Science*, 2018.

[102] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[103] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[104] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[105] D. Nadeau, P. D. Turney, and S. Matwin, "Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity," in *Conference of the Canadian society for computational studies of intelligence*, pp. 266–277, Springer, 2006.

[106] S. Zhang and N. Elhadad, "Unsupervised biomedical named entity recognition: Experiments with clinical and biological texts," *Journal of biomedical informatics*, vol. 46, no. 6, pp. 1088–1098, 2013.

[107] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd, "spaCy: Industrial-strength Natural Language Processing in Python," 2020.

[108] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[109] R. Řehůřek and P. Sojka, "Software framework for topic modelling with large corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, (Valletta, Malta), pp. 45–50, ELRA, May 2010.

[110] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *arXiv preprint arXiv:1607.04606*, 2016.

[111] L. Chen and A. Moschitti, "Learning to progressively recognize new named entities with sequence to sequence models," in *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 2181–2191, 2018.

[112] E. Rosvall, "Comparison of sequence classification techniques with bert for named entity recognition," 2019.

[113] S. L. Ingólfsdóttir, *Named entity recognition for Icelandic: annotated corpus and neural models*. PhD thesis, Reykjavik University, 2020.

[114] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," *University of Pennsylvania*, 2001.

[115] J. Teixeira, L. Sarmento, and E. Oliveira, "A bootstrapping approach for training a ner with conditional random fields," in *Portuguese Conference on Artificial Intelligence*, pp. 664–678, Springer, 2011.

[116] Q. Wei, T. Chen, R. Xu, Y. He, and L. Gui, "Disease named entity recognition by combining conditional random fields and bidirectional recurrent neural networks," *Database*, vol. 2016, 2016.

[117] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.

[118] B. Haak, "Information extraction from homicide-related dutch texts using bert," *Jheronimus Academy of Data Science*, 2020.

[119] M. E. Peters, S. Ruder, and N. A. Smith, "To tune or not to tune? adapting pretrained representations to diverse tasks," *arXiv preprint arXiv:1903.05987*, 2019.

[120] I. Beltagy, K. Lo, and A. Cohan, "Scibert: A pretrained language model for scientific text," *arXiv preprint arXiv:1903.10676*, 2019.

[121] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "Biobert: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

[122] Y. Gu, R. Tinn, H. Cheng, M. Lucas, N. Usuyama, X. Liu, T. Naumann, J. Gao, and H. Poon, "Domain-specific language model pretraining for biomedical natural language processing," *arXiv preprint arXiv:2007.15779*, 2020.

[123] Y. Elazar, N. Kassner, S. Ravfogel, A. Ravichander, E. Hovy, H. Schütze, and Y. Goldberg, "Measuring and improving consistency in pretrained language models," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 1012–1031, 2021.

[124] W. Tai, H. Kung, X. L. Dong, M. Comiter, and C.-F. Kuo, "exbert: Extending pre-trained models with domain-specific vocabulary under constrained training resources," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pp. 1433–1439, 2020.

[125] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015.

[126] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[127] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *arXiv preprint arXiv:1808.06226*, 2018.

[128] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[129] N. Kitaev, Ł. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," *arXiv preprint arXiv:2001.04451*, 2020.

[130] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[131] R. G. Londoño, S. Wlodarczyk, M. Arman, F. Bugiotti, and N. B. Seghouani, "Weakly supervised named entity recognition for carbon storage using deep neural networks," in *International Conference on Discovery Science*, pp. 227–242, Springer, 2022.