



Norwegian University of
Science and Technology

Automated Information Extraction in Natural Language

Josef Emil Horgdal Fritzner

Master of Science in Mechanical Engineering

Submission date: June 2017

Supervisor: Amund Skavhaug, MTP

Norwegian University of Science and Technology
Department of Mechanical and Industrial Engineering

Preface

This thesis, “Automated Information Extraction in Natural Language”, is a feasibility study of utilizing machine learning to automatically process natural language. It is written as part of the Master of Science degree, Mechanical Engineering at NTNU, during the spring semester of 2017.

The project was undertaken at the request of the company AVO Consulting. The idea for the project was brought up as part of AVO Consulting’s current research in natural language processing, and the research question was formulated together with AVO Consulting manager, Preben Høeg Hafsaas. Early in the semester, two weeks of robotic process automation software training (Blueprism) were completed at AVO Consulting’s offices in Oslo. During these weeks, experimentation was also conducted in the natural language processing platform, Watson Knowledge Studio. In addition to this, one workshop on Watson Knowledge Studio was attended on the 29th of March, which was held by IBM’s solution architect, Mark Rice. This laid the foundation for the case work.

I would like to express my gratitude to my supervisor, Professor Amund Skavhaug, for the useful comments, remarks and engagement throughout the learning process of this master thesis. Furthermore, I would like to thank the employees of AVO Consulting for introducing me to the topic, as well for the cooperation on the way. Finally, my parents, sister and girlfriend deserve a particular note of thanks; your continuous encouragement throughout my years of study kept me motivated.

This thesis is intended for those who wish to get an insight in the current field of cognitive natural language processing. I hope you enjoy your reading.

Trondheim, 11.06.17

A handwritten signature in black ink, reading "Josef E Fritzner". The signature is written in a cursive, flowing style.

Josef Emil Fritzner

Executive Summary

The field of service automation is progressing rapidly, and increasingly complex tasks are being automated by robots. An area in service automation that has received a lot of attention is Natural Language Processing (NLP). In today's digital age, enormous amounts of data are produced, and most of this data is in a so-called unstructured form, including text in natural language. Such text holds information that can be very valuable to businesses, but is seen as time consuming and difficult to analyze in the business' perspective. NLP is the computerized approach to analyzing and representing human language, and can be utilized for automatic extraction of relevant information from text.

Since the statistical revolution in the late 1980s, much of the research in NLP has been based on machine learning. Machine learning enables an NLP systems to automatically learn patterns of language from text samples, and recognize these patterns in new, unseen text. Machine learning is particularly applicable in language processing, where the rules often are ambiguous and difficult to process with manual coding. Most modern systems for NLP use supervised learning to train the machine learning component. This means that relevant information in text must be manually labeled in order to use the text to train a machine learning model. By selecting which information is relevant in text samples from a specific domain, the model can be customized to achieve the goal in that domain.

The aim of this thesis is to provide insight into how a modern NLP system works, its limitations and potential applications. The theoretical aspect of using machine learning in NLP is presented with focus on the task of extracting information from text in natural language. A case is introduced with a design of an application for extracting information from emails from the shipping industry. The machine learning model is trained with emails which have been labeled for names of ships, contract types, and ports and dates for chartering. Two machine learning models are trained in an available NLP system called "Watson Knowledge Studio". The first model is able to recognize and label some of the variables in new emails, but is relatively inaccurate. The results are improved in the second version of the model by following the practices recommended in the discussed theory.

The results of the case confirm the necessity of large amounts of labeled data for robust training of a machine learning model. The results also indicate that the text should have linguistic structure to a certain degree in order for the underlying rules for grammar in the NLP system to be exploited for optimal processing. If the language in the domain does not have sufficient linguistic structure, other rule-based methods should also be considered, in addition to the machine learning method. A hybrid approach may be the best solution in these cases.

The demonstrated case shows how the customization of a machine learning model can be based on domain knowledge, rather than coding-skills, when the system uses supervised learning to train the model. This means that the system can reach a larger target group, with more people able to take advantage of machine learning in NLP. Based on this, it may be assumed that this method will have a key role in the future of NLP. With development of support for the Norwegian language, AVO Consulting can potentially benefit from the advantages of using supervised machine learning in NLP.

Sammendrag

Service-automatisering som forskningsfelt har utviklet seg mye de siste årene, og stadig mer komplekse oppgaver kan automatiseres av roboter. Et område innen service-automatisering som har fått mye oppmerksomhet er Natural Language Processing (NLP). I dagens digitale alder produseres det enorme mengder data, og mesteparten av denne dataen er i en såkalt ustrukturert form, inkludert tekst i naturlig språk. Slik tekst inneholder informasjon som kan være svært verdifull for firmaer, men er tidkrevende og vanskelig å analysere, sett fra firmaets perspektiv. NLP handler om å bruke beregningsmetoder for å analysere og representere naturlig språk, og kan blant annet brukes til å automatisk hente ut relevant informasjon fra tekst.

Siden den statistiske revolusjonen på slutten av 1980-tallet, har mye av forskningen i NLP dreid seg om maskinlæring. Maskinlæring gjør det mulig for et NLP system å automatisk lære seg språktrekk fra teksteksempler, og gjenkjenne disse trekkene i annen tekst. Maskinlæring egner seg svært godt for prosessering av språk, hvor reglene ofte er tvetydige og vanskelige å behandle med manuell koding. I de fleste moderne systemer for NLP blir opplæringen av maskinlæringsmodellen veiledet. Det vil si at relevant informasjon i teksten må markeres manuelt for at teksten skal kunne brukes til å trene opp en maskinlæringsmodell. Ved å markere hvilken informasjon som er relevant i teksteksempler fra et spesifikt domene kan modellen tilpasses for å oppnå målet i det gitte domene.

Målet med denne avhandlingen er å gi et innblikk i hvordan et moderne NLP system fungerer, dets begrensninger og potensielle bruksområder. Det teoretiske rammeverket rundt bruken av maskinlæring i NLP presenteres, og fokuserer på utvinning av informasjon fra tekst i naturlig språk. En case introduseres med et design av en applikasjon for å hente ut informasjon fra e-mailer fra skipsmeglingsindustrien. Maskinlæringsmodellen trenes opp med e-mailer som er markerte for navn på skip, kontrakt-type, samt havn og dato for utleie. Det utvikles to modeller i et tilgjengelig NLP system kalt "Watson Knowledge Studio". Den første modellen er i stand til å gjenkjenne og markere noen av variablene i nye e-mailer, men er relativt upresis. Resultatene forbedres i den andre versjonen av modellen ved å følge praksiser anbefalt i den diskuterte teorien.

Resultatene av casen bekrefter nødvendigheten av store mengder markert data for robust trening av maskinlæringsmodellen. Resultatene tyder også på at teksten bør ha en viss språklig struktur for at de underliggende reglene for grammatikk i NLP systemet skal kunne benyttes for optimal prosessering. Om språket i domenet er av en lav språklig struktur bør også andre regelbaserte metoder vurderes fremfor maskinlæring. En hybrid tilnærming kan være den beste løsningen i disse tilfellene.

Den gjennomførte casen demonstrerer at tilpasningen av en maskinlæringsmodell kan baseres på kunnskap om domenet fremfor kodeforståelse når opplæringen av modellen er veiledet. Dette gjør at teknologien kan nå en større målgruppe, med mange flere i stand til å dra nytte av maskinlæring i NLP. Basert på dette kan det antas at denne metoden for maskinlæring vil spille en sentral rolle i fremtiden for NLP. Ved utvikling av støtte for det norske språket vil antageligvis maskinlæring i NLP være noe som AVO Consulting kan dra stor nytte av i sin virksomhet.

Contents

| | |
|--|----------|
| Preface | i |
| Executive Summary | iii |
| Sammendrag | v |
| 1 Introduction | 1 |
| 1.1 Background | 1 |
| 1.2 Problem Description with Objectives | 3 |
| 1.3 Limitations | 3 |
| 1.4 Approach | 4 |
| 1.5 Structure of the Thesis | 4 |
| 2 Theoretical Framework | 5 |
| 2.1 Unstructured Data | 5 |
| 2.2 Natural Language Processing | 7 |
| 2.2.1 Syntactic Ambiguity | 8 |
| 2.2.2 Brief History of Natural Language Processing | 9 |
| 2.3 Machine Learning | 10 |
| 2.3.1 Ways of Learning | 11 |
| 2.4 Information Extraction | 13 |
| 2.5 Algorithms in Information Extraction | 15 |
| 2.5.1 Naive Bayes | 18 |
| 2.6 Evaluation in Information Extraction | 21 |
| 2.6.1 The Test Set | 21 |
| 2.6.2 Precision and Recall | 22 |

| | |
|---|-----------|
| 3 Case: IE from Shipping Industry Emails | 23 |
| 3.1 Motivation | 23 |
| 3.2 Method | 24 |
| 3.2.1 Unstructured Information Management Architecture (UIMA) | 25 |
| 3.2.2 Statistical Information and Relation Extraction (SIRE) | 27 |
| 3.2.3 Analysis of Statistics | 30 |
| 3.2.4 Improvement of ML Model Performance | 30 |
| 3.3 Procedure | 32 |
| 3.3.1 Defining the Type System | 34 |
| 3.3.2 Rule Creation | 35 |
| 3.3.3 Annotating in the Ground Truth Editor | 41 |
| 3.4 Results | 44 |
| 3.4.1 Statistics of ML Model 1.0 | 44 |
| 3.4.2 Annotating with the ML Model | 48 |
| 3.4.3 Retraining the ML Model | 51 |
| 3.4.4 Statistics of ML Model 1.1 | 53 |
| 3.4.5 Decoding Results | 56 |
| 4 Summary | 59 |
| 4.1 Discussion | 60 |
| 4.2 Conclusions and Further Work | 64 |
| A Acronyms | 67 |
| B Watson Knowledge Studio Documentation | 69 |
| B.1 Registration | 69 |
| B.2 Browser Requirements | 69 |
| B.3 Assembling a Team | 70 |
| B.4 Creating a Project | 74 |
| Bibliography | 81 |

Chapter 1

Introduction

1.1 Background

Companies are achieving productivity gains by using Robotic Process Automation (RPA) to perform routine, rule-based service processes. These software robots, often referred to as just “robots”, can perform repetitive tasks previously carried out by humans, so that humans can focus on more challenging tasks that require creativity and empathy. If implemented well, RPA can result in high-performing human-robot teams, in which robots and human employees complement one another. Such teams amplify distinctive human strengths while simultaneously enabling large economic gains. In a study of early adopters of RPA systems carried out by Lacity and Willcocks (2016), participants report a return on investment of 30% or more during the first year of implementation. In addition to the financial benefits, robots can increase the speed, quality and availability of service. The reason for this is that robots can execute structured tasks quickly and accurately, without the need to rest or sleep. The robotic workforce is also easily scalable to the work-load, with humans filling in the gaps that require on-the-fly problem solving, and hands-on customer care.

RPA is currently being used to effectively automate lower-level tasks that involve structured data and defined outcomes. However, the field of service automation is progressing rapidly, and as more and more companies see the value of RPA and adapt it in their processes, the demand for automation of higher-level tasks increases. Higher-level tasks involve unstructured data and typically require cognitive automation tools to automate. It has been estimated that 80 percent

of business-relevant information originates in unstructured form, primarily text (Shilakes and Tylman, 1998). Unstructured data is vastly underutilized due to challenges of analyzing and using it. Exploiting this source of information is an essential component in ensuring enterprise competitiveness. In order to further explain what is meant by unstructured data, a detailed definition is given in Section 2.1.

One of the tools that is effectively being used to automate higher-level tasks is called Machine Learning (ML). Implementation of ML allows the computer to learn from sets of data, without being explicitly programmed, enabling what is known as cognitive automation.

This thesis includes research in one of the applications of cognitive automation, namely Natural Language Processing (NLP). Widely used literature on NLP include Jurafsky (2000) and Manning et al. (1999, 2008). Research on the subject is regularly published in the annual proceedings of the Association of Computational Linguistics (ACL) and its European counterpart EACL.

Information Extraction (IE) is an area of NLP that has been gaining increasing recognition for its capability and applicability in many tasks concerned with analysis of unstructured data. Publications from the Message Understanding Conferences (1978-1997) describe the first rule-based approaches to IE (DeJong, 1982; Appelt et al., 1993). The publications; Leek (1997), McCallum et al. (2000) and Lafferty et al. (2001) describe some of the later developed, trainable algorithms of ML, which are currently used in many IE systems.

Based on the publications mentioned and other resources, this thesis describes the theory of NLP and the subcategory of IE. The remaining problems to be researched is how ML can be implemented in a system for IE, how well the resulting system performs, its applications, and how the ML approach compares to rule-based approaches in NLP.

1.2 Problem Description with Objectives

This thesis is a feasibility study of the practicality of using modern systems for natural language processing. Its main goal is to give AVO Consulting and the department of mechanical and industrial engineering at NTNU insight in the theory and application of machine learning in natural language processing. This paper presents:

- Relevant background.
- A proposal of design of an example system relevant to the firm's processes.
- Implementation of proposed system.
- Experimentation with available data.
- An assessment of the quality of obtained system.

1.3 Limitations

The development of cognitive robots able to perform increasingly more complex tasks has led to widespread skepticism and fears about how many types of employment will fare in the future. The media is fueling the fear with headlines like “Robots will take our jobs - and we're not ready for it” from the Guardian (Shewan, 2017), while some headlines like “Automation can actually create more jobs” from the Wall Street Journal (Mims, 2016) present a more optimistic outlook on the future. What most people agree on is that the labor market will transform due to the development of artificial intelligence. Today's rapid adaptation of robots across several industries suggests that the transition is inevitable. It is important to consider how humans and robots can cooperate in a labor market with an increasing number of robots and applicable tasks. This dilemma is not a focus of this thesis, due to the scope of the study.

The implementation of the proposed design of a system is limited by the time and data available. The system implemented in Chapter 3 is mainly designed to demonstrate how the technology can be used in a real scenario.

1.4 Approach

Publications on the subjects of NLP and ML are studied, and the theory considered relevant to the case is presented in the thesis. A design of an example system is proposed and implemented in IBM's service; Watson Knowledge Studio. This platform incorporates ML, as well as rule-based tools for processing of natural language, and is suitable for the demonstration purposes of the case. The proposed system is implemented for processing of emails from the shipping industry. This data is chosen, based on the supported languages of the application, and the relevance to AVO Consulting's processes. Two different approaches are compared, and the quality of the obtained system is assessed by evaluation techniques described in the theory.

1.5 Structure of the Thesis

The rest of the thesis is organized as follows. Chapter 2 gives an introduction to some of the terms and concepts in ML and NLP. The chapter delimits one of the areas in NLP, namely IE. Relevant algorithms for training ML models for IE and typically used methods for analyzing the performance of the models are described. Chapter 3 outlines the motivation behind the case work, and presents the procedure of implementation and the resulting system. In Chapter 4 the results of the case are summarized and discussed, and conclusions are drawn on the basis of the discussed theory.

Chapter 2

Theoretical Framework

2.1 Unstructured Data

Unstructured data refers to information that either does not have a pre-defined data model, or is not organized in a pre-defined manner. Unlike structured data, it is not arranged in a relational database system and is therefore not readily searchable by simple, straightforward search operations. Examples of unstructured data may include books, journals, emails, metadata, images, audio and video. The data can be categorized as either textual or non-textual. The topic of this thesis concerns the unstructured textual data of natural language.

The term, unstructured data, may seem to imply complete randomness with no inherent form. This is due to imprecision of the term. In fact, unstructured data incorporates structure to a certain degree in one of three ways:

1. Structure can be implied, even if not formally defined.
2. Data with some form of structure may still be characterized as unstructured, if its structure is not helpful for the task at hand.
3. Unstructured data might have some structure, sometimes referred to as semi-structured, or even be highly structured, but in ways that are unanticipated or unannounced.

Natural language incorporates structure in the form of grammatical rules. However, these rules often have accepted variations, and the structure of language is often influenced by non-

homogeneous grammatical rules and errors of individuals. Probabilistic, or non-categorical models provide suitable methods for describing structure in variable systems like in natural language. The probabilistic models of ML are described in Section 2.5.

Unstructured data is less understood and utilized by companies than the structured counterpart. Unstructured data can hold information that may be useful to businesses. In a report from the Aberdeen Group (Michael, 2016), businesses using unstructured data were twice as likely to be satisfied with their data quality and usability.

Unstructured data can be valuable for decision-making in many processes, such as customer-centric processes, where decisions are made on the basis of interactions between different parts. Decisions requiring unstructured data typically involve time consuming, manual capture work. To confidently automate processes that involve this type of decision-making, information has to be extracted from unstructured data. Tasks involving unstructured data are best handled by cognitive automation tools. The different service characteristics in each of the two realms of automation are made evident in Table 2.1.

Table 2.1: Service Characteristics (Lacity and Willcocks, 2016).

| | Realm of robotic process automation | Realm of cognitive automation |
|----------------|---|--|
| Data | Structured | Unstructured |
| Process | Rules-based | Inference-based |
| Outcome | Single correct answer | Set of likely answers |
| | Robotic process automation tools are designed to be used by subject matter experts to automate tasks that use rules to process structured data, resulting in a single correct answer - in other words, a deterministic outcome | Cognitive automation tools are designed to be used by IT experts to automate tasks that use inferences to interpret unstructured data, resulting in a set of likely answers, as opposed to a single answer - in other words, a probabilistic outcome. |

2.2 Natural Language Processing

Natural Language Processing (NLP) is the computerized approach to analyzing human language. A detailed definition of the term is proposed in Liddy (2001);

Natural Language Processing is a theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications. (p. 2)

Elements of this definition can be further detailed. Firstly, 'naturally occurring texts' includes both oral and written language. This thesis deals with text in the form of written language, but similar techniques and theories can be applied to the analysis of oral language by utilizing a system for speech recognition (Allen, 2003a).

'Human-like language processing' reveals that NLP is considered a discipline within Artificial Intelligence (AI). While some of the functions of NLP depend on a number of other disciplines, it is appropriate to consider NLP an AI discipline, as it strives for human-like performance.

'For a range of tasks or applications' points out that NLP is not usually considered a goal in and of itself, but rather the means for accomplishing a particular task. NLP is utilized for several tasks, including Information Extraction (IE), Machine Translation (MT), Question-Answering, etc.

2.2.1 Syntactic Ambiguity

NLP is considered a complex field in computer science. To understand human language is to understand not only the words, but how the words are linked together to create meaning. The ambiguity of language is what makes NLP a difficult problem for computers to master. Natural language involves five forms of ambiguity, according to Allen (2003b):

- Simple lexical ambiguity (e.g. “duck” can be a noun (the animal) or a verb (to avoid something thrown)).
- Structural or syntactic ambiguity (e.g. in “I saw the man with a telescope,” the telescope might be used for the viewing or might be held by the man being observed).
- Semantic ambiguity (e.g. “go” as a verb has well over 10 distinct meanings in any dictionary).
- Pragmatic ambiguity (e.g. “Can you lift that rock?” may be a yes/no question or a request to lift the rock).
- Referential ambiguity (e.g. in “Jack met Sam at the station. He was feeling ill. . . ,” it is not clear who is ill, although the remainder of the sentence might suggest a preferred interpretation).

In the last decades, the field of NLP has been progressing rapidly. Current NLP systems tend to implement modules for processing of more forms of ambiguity than before. This can be attributed to the increased availability of large amounts of electronic text, increased computational speed and memory, and the advent of the internet.

2.2.2 Brief History of Natural Language Processing

The earliest research in NLP dates back to the late 1940s, with translation being the first computer-based application related to natural language. The idea was called Machine Translation (MT), and was brought to general notice in Weaver's memorandum, simply called "Translation", in 1949. The early MT systems used dictionary-lookup for appropriate words for translation, and then reordered the words to fit the word-order rules of the target language. This type of system did not take any forms of ambiguity of language into account, and therefore produced relatively poor results. The apparent failure made researchers realize that the task was a lot harder than anticipated, and relatively little further research in MT was conducted over the next few decades.

In 1988, at the Second TMI conference at Carnegie Mellon University, IBM's Peter Brown presented a new approach to MT. Instead of using linguistic rules to translate, this system used statistical models whose parameters were derived from the analysis of text corpora (Somers, 1998). By looking at several possible solutions and assigning probabilities, statistical approaches succeeded in dealing with many generic problems in computational linguistics, such as word sense disambiguation (WSD), part-of-speech (POS) identification, parsing, etc., and became the standard throughout NLP (Liddy, 2001).

The "statistical revolution" in NLP laid the foundation for the newest technological advancement in NLP, namely Machine Learning.

2.3 Machine Learning

Modern NLP systems typically incorporate Machine Learning (ML) algorithms. Instead of hand-writing large sets of rules for NLP, modern systems can rely on ML to automatically process text. The algorithms are trained to learn the patterns of language from a set of examples, like books, news articles or other types of text corpus. Systems based on ML algorithms have many advantages over systems based on hand-produced rules:

- Automatic learning procedures can make use of statistical inference algorithms. These algorithms are able to deal with unfamiliar input, like words or structures that have not yet been seen, and erroneous input, like misspelled or accidentally omitted words. While this also is possible to achieve by hand-writing rules that make soft decisions, it is considered extremely difficult, error-prone and time-consuming.
- The learning procedure of ML automatically focuses on the most commonly occurring cases, whereas when writing rules by hand it is often not obvious where the effort should be directed.
- Systems based on automatically learning rules can be made more accurate by supplying more input data in contrast to systems based on hand-written rules, which can only be made more accurate by increasing the volume and complexity of rules. This makes ML systems more scalable than rule-based systems.
- ML approaches are generally more robust and domain-independent than rule-based approaches due to the knowledge of the corpus required to write robust rules (Joachims, 2002).

Data gathering and preparation is a vital part of the ML process, since the data is used to train the model and thus determines how the model will perform. It is important to have a data set that accurately represents the language of the corpus that is to be processed. The set for training will typically require some form of curating, for example, by removing text in different languages or parts of the text that are unique to the training set. The final selection of examples is used as input to train the ML model. The output of the model will perform as accurately as

possible on the training set, and if the set is representative, the model is able to perform accurately on sets of new, unseen data. Overfitting can occur when a model learns the detail and noise of the training data to the extent that it negatively impacts the performance of the model on new data (Hawkins, 2004). Overfitting can be avoided by testing the model's ability to generalize by evaluating its performance on a set of data not used for training, commonly referred to as test data. Underfitting occurs when the model cannot capture the underlying trends of the data, and often results in an extensively simple model. Figure 2.1 shows the correlation between the prediction error and the complexity of a model. The global minimum for validation error indicates the optimal degree of training; where the best predictive model is obtained.

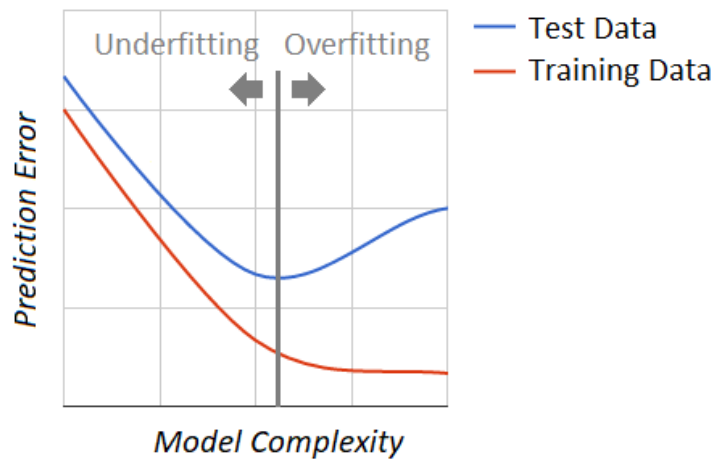


Figure 2.1: Learning Trade-offs

2.3.1 Ways of Learning

There are essentially two ways of training an ML model for NLP; unsupervised learning and supervised learning. Unsupervised learning is the task of inferring a function to describe a hidden structure in unlabeled data. The fact that the data is unlabeled means that data used for training has no classification, and that the resulting model can not be evaluated against a manually labeled set. Unsupervised learning is considered a complex task in data science, but if implemented successfully in NLP, it can mimic the way humans learn language at early ages. Latent Dirichlet Allocation (Blei et al., 2003), and Skip-Gram models (Mikolov et al., 2013) are examples of unsupervised learning algorithms that have been shown to work well for problems

in NLP. Generally, the algorithms used for unsupervised learning involve grouping objects that share similarities into clusters to discover patterns. How the scores for similarity are applied to objects is what separates the different algorithms.

Given the abundance of labeled text data and the challenge of labeling it, unsupervised learning techniques, and semi-supervised learning techniques (using a combination of labeled and unlabeled data) have gained considerable attention in the field of NLP. The techniques can model patterns which are not obvious to people, such as the ambiguous form of natural language. However, despite the great deal of research in unsupervised learning, practical ML approaches to NLP are almost exclusively supervised (Klein, 2005). When it comes to domain-specific language in the industry, supervised learning is the preferred method. In contrast to unsupervised learning, supervised learning relies on hand-labeled corpora for training. In return, this labeled data can be used as a "gold standard" for measuring accuracy of the labels generated by the ML model. New gold standards can be defined by labeling domain-specific data, which makes the system more adaptable than an unsupervised one. Huang et al. (2014) states that supervised NLP tasks with sufficiently labeled, domain-specific training data yield state-of-the-art results.

Popular algorithms used in supervised learning for NLP include logistic regression, Naïve Bayes, Support-Vector Machines (SVMs), k-Nearest Neighbors and Markov Models. These algorithms all involve some form of statistical inference, which is a crucial part of capturing linguistic knowledge. The key advantage of probabilistic models is their ability to solve the many kinds of ambiguity problems; almost any problem in NLP can be recast as "given N choices for some ambiguous input, choose the most probable one" (Jurafsky, 2000). The algorithms for supervised learning generate features from the input data, and soft, probabilistic decisions are made based on attaching real-valued weights to each feature. Thus, the model can express the relative certainty of several possible answers, and chose the most probable.

2.4 Information Extraction

The task of Information Extraction (IE) is to derive factual structured information from unstructured data. This involves presenting relevant parts of unstructured data in a format that is more easily machine readable. For instance, consider as an example the extraction of information on traffic incident reports, where one is interested in identifying the vehicles involved, the type of incident, and the time and date of the incident. By labeling words (also referred to as annotating) in the text which are significant to the reader, the relevant information can readily be extracted. Figure 2.2 shows an excerpt of a traffic incident report and the structured information derived from that text.

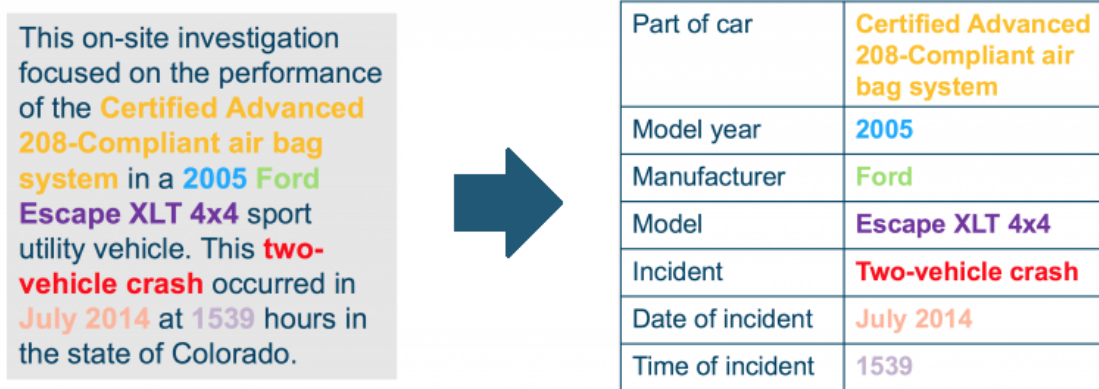


Figure 2.2: Information extraction from a traffic incident report (Sinha and Boyd, 2016).

Nowadays, a significant part of information is transmitted through unstructured, digital documents, e.g., online news, government documents, corporate reports, medical records, and social media communication. Extracting information from these documents requires tedious search operations. Recent advances in IE provide dramatic improvements in the automatic conversion of the flow of raw textual information into structured data. For this reason, IE is increasingly being deployed in commercial applications in several domains, including the financial, legal, medical and security domain. Furthermore, IE can constitute a core component in many other NLP applications, such as Machine Translation, Text Summarization, Question Answering, etc. (Piskorski and Yangarber, 2013). There are several subtasks of IE required to derive structured information. Typical subtasks include:

- **Named Entity Recognition (NER)** is a subtask of IE which involves locating and classifying named entities in natural language into pre-defined categories. The procedure consists of taking an unannotated block of text and producing an annotated block of text that highlights the names of entities. A named entity can include every part of a text that is easily distinguishable, such as names of persons, organizations and locations, and expressions of dates, quantities, etc.
- **Co-reference Resolution (CO)** requires the identification of multiple mentions of the same entity in a text. In the sentence “John said he would come”, "John" and "he" refer to the same person, and should be coreferenced for this entity type. CO is an important step for natural language understanding for higher level tasks, like IE.
- **Relation Extraction (RE)** is the task of detecting and classifying relationships between entities identified in the text. For example, in the sentence “This two-vehicle crash occurred in July 2014 in the state of Colorado” a relationship between the two mentions "two-vehicle crash" and "the state of Colorado" may be classified as the location of the incident. Note that the set of relations available in a task is predefined and fixed as part of the specification of the task.
- **Part-of-Speech (POS) tagging** is the task of assigning lexical categories such as verb, noun or adjective to words in a sentence. The POS tag of a word depends on both its definition and its context, i.e., its relationship with adjacent and related words. POS tagging is commonly used as an intermediate processing stage for the task of IE. For languages with substantial amounts of labeled data such as English, the performance of POS tagging has reached very high levels for tasks like IE (Abend et al., 2010).
- **Parsing** is the process of identifying the grammatical structure of a sentence. Parsing is done by converting a sentence into a tree structure whose leaves hold the POS tags to show how the words are combined to make the sentence. Statistical parsers, tools that automatically parse sentences, have long been used as the backbone for IE (Chieu et al., 2003; Miller et al., 2000).

2.5 Algorithms in Information Extraction

The subtasks of IE, mentioned in the previous section, are concerned with the problem of classification. NER, CO and RE are needed to create a solid structure of entities from the text, while POS tagging and Parsing are needed to find the linguistic structure of the text. Algorithms of an ML model can be trained to automatically perform the relevant subtasks and classify the contents of documents. To train the algorithms, relevant entities and linguistic structure are manually labeled in a set of documents. The labeled set is then allocated for training the algorithms. IE systems are typically used in domains with refined tasks, which means that the corpus requires labels specific to the domain-language. By manually labeling the corpus, it contains the correct labels for each input, and can be used as a gold standard for training the ML algorithms. This process is called supervised classification, and its framework is shown in Figure 2.3.

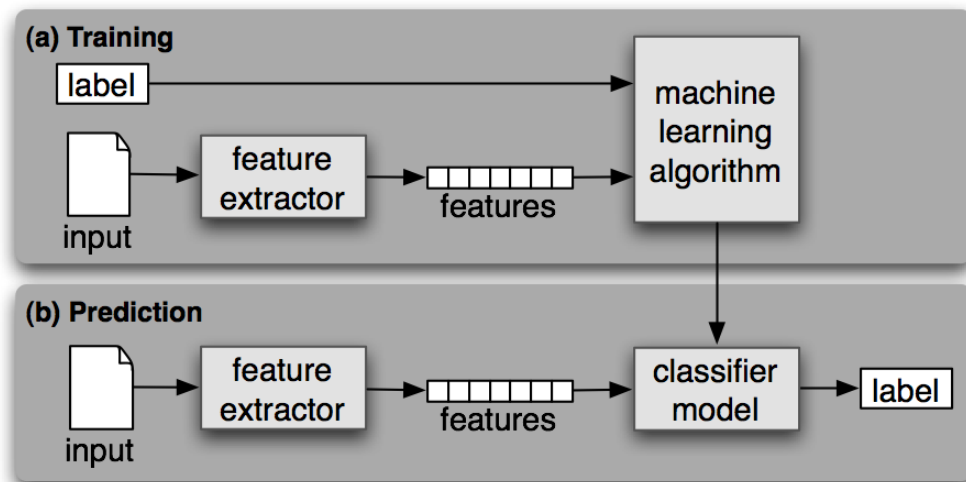


Figure 2.3: Supervised Classification (Bird et al., 2009).

(a) During training, features from the training set and their associated labels are fed into the ML algorithm to generate a model. (b) During prediction, features from the new data is fed into the model, which generates predicted labels.

The feature extractor uses a function that can take account of relations between both data and labels, and can be any real value. The most commonly applied feature function is the indicator function, which outputs 0 or 1 depending on absence or presence of a feature. The features can for instance be capitalization of words, word-ending, or whether the word has been observed with a label in the training data.

In most IE systems, the ML model is trained using sequence classifiers. These algorithms have the advantage of being able to classify several inputs simultaneously. The most common sequence classifiers for IE tasks are Hidden Markov Model (HMM), Maximum-Entropy Markov Model (MEMM), and Conditional Random Field (CRF) (McCallum et al., 2000). These sequence classifiers are algorithms which count occurrences of features and assign probabilities for certain sequences. For example, in the case of POS tagging, the word following “the” may be observed as a noun 40% of the time, an adjective 40% of the time, and a number 20% of the time in the training data. By learning this, the system can decide that the word “can” in the context “the can” is more likely a noun than a verb. This is an example of a sequence of two words, but more advanced, higher-order algorithms learn the probabilities not only of pairs, but triples or even larger sequences. In the case of NER, the likelihood of, for example, the name of a person to appear in a sequence of words can be calculated using these sequence classifiers. The relationships between some of the algorithms commonly used in IE are illustrated in Figure 2.4.

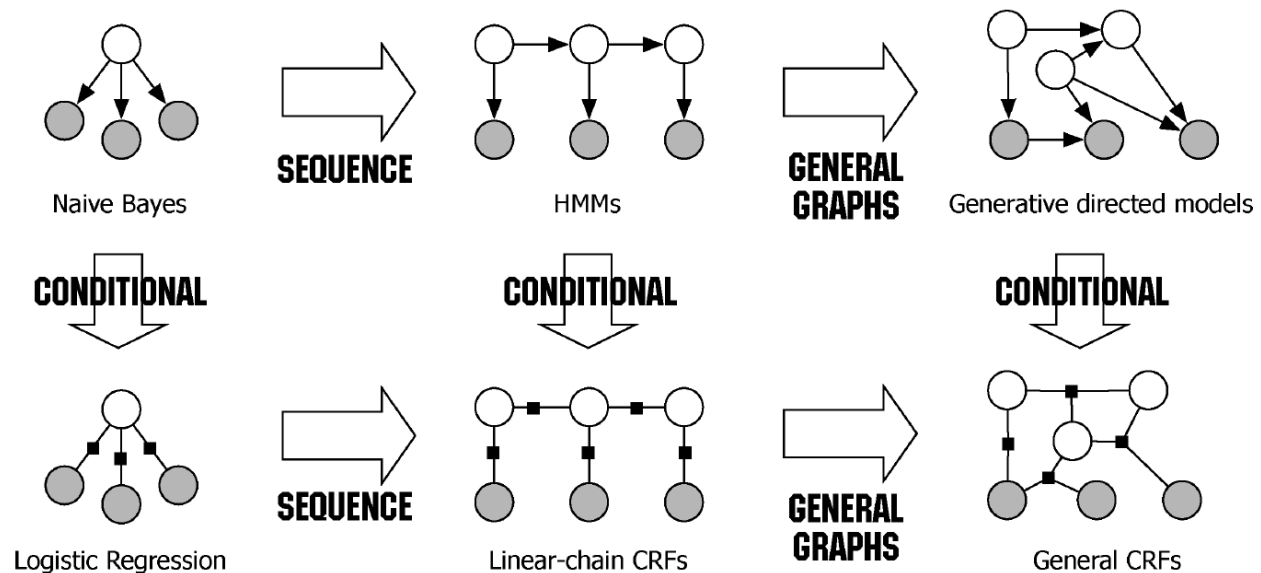


Figure 2.4: Diagram of the relationships between naive Bayes, logistic regression, HMMs, linear-chain CRFs, generative models, and general CRFs (Sutton and McCallum, 2006).

One difference between the three most commonly used algorithms in IE is that HMM is a generative sequence model, while MEMM and CRF are discriminative sequence models. This means that if X is the feature and Y is the label, HMM finds parameters to maximize the probability $P(X, Y)$, while MEMM and CRF find parameters to maximize the probability $P(Y|X)$. Unlike HMM, MEMM and CRF do not need to model the probability $P(X)$, and therefore do not assume that the observed features X_i are conditionally independent. This allows for many more types of features to be used in MEMM and CRF than HMM. HMM and CRF have the advantage of taking future observations into account when labeling features X_i , while MEMM makes decisions at each step, and can potentially suffer from the "label bias problem". The label bias makes the algorithm ignore observations in states with low-entropy transition distribution. CRF is a globally normalized, discriminative sequence model which incorporates the advantages from both HMM and MEMM, but can in return suffer from a much more expensive training process. Naive Bayes is the non-sequential counterpart to HMM and is described in Section 2.5.1 to provide a general understanding of how classification algorithms work.

2.5.1 Naive Bayes

The naive Bayes model is one of the most important algorithms for text classification and was first applied to IE in Freitag (1997, 2000). Naive Bayes methods are based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features. It essentially uses a bag-of-words representation of text, in which the ordering of the words appearing in the text is ignored. If a word does not occur in a document, the corresponding element in the bag-of-words representation of the document is typically zero. Otherwise, the element is assigned a real-valued weight that indicates the importance of the word in the document.

In an ML classification problem with a given class variable c (also referred to as a label), the main aim of the naive Bayes algorithm is to calculate the conditional probability of an object with a feature vector x_1, x_2, \dots, x_n belonging to that particular class. Bayes' theorem states the following relationship:

$$P(c|x_1, x_2, \dots, x_n) = \frac{P(x_1, x_2, \dots, x_n|c)P(c)}{P(x_1, x_2, \dots, x_n)}. \quad (2.1)$$

In order to make the estimate $P(x_1, x_2, \dots, x_n|c)$, naive Bayes makes the simplifying assumption that each occurrence of a feature in a document can be considered as evidence of class membership, independent of any other feature occurring in the same document. With the naive independence assumption, it is stated that

$$P(x_1, x_2, \dots, x_n|c) = \prod_{k=1}^n P(x_k|c), \quad (2.2)$$

and since $P(x_1, x_2, \dots, x_n)$ is constant given the input, the following classification rule can be derived from Bayes' theorem:

$$P(c|x_1, x_2, \dots, x_n) \propto \prod_{k=1}^n P(x_k|c)P(c) \quad (2.3)$$

The Maximum A Posteriori (MAP) estimation can then be used to estimate $P(c)$ and $P(x_1, x_2, \dots, x_n|c)$ from the training data and assign the most probable class c_{MAP} using the formula:

$$c_{MAP} = \arg \max_{c \in C} P(x_1, x_2, \dots, x_n|c)P(c) \quad (2.4)$$

The estimate of $P(c)$ gives prior probability of each label, which is determined by checking frequency of each label in the training set. $P(x_1, x_2, \dots, x_n | c)$ gives the contribution from each feature. When combined, the likelihood estimate for each label can be calculated. The label whose likelihood estimate is the highest is then assigned to the input value. This is demonstrated in a topic classification example illustrated in Figure 2.5, which uses the naive Bayes classifier to predict one of three topics for documents; automotive, sports or murder mystery.

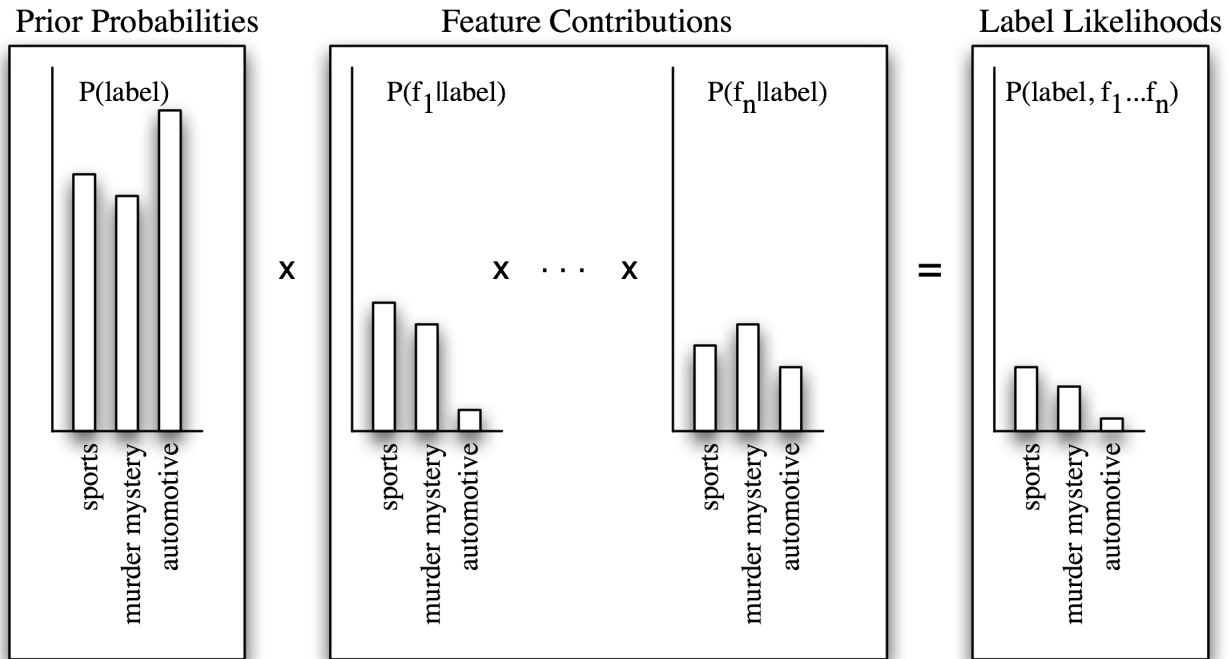


Figure 2.5: Calculating label likelihoods in topic classification with naive Bayes (Bird et al., 2009).

In the example training corpus, most documents are automotive, so the prior probability predicts the "automotive" label for new documents. Features in the new documents also contribute to the final likelihood estimate of the document label. The feature contribution is given by the probability of features occurring in each of the three types of documents in the training corpus. For instance, the word "run" may be defined as a feature that occurs in 12% of the sports documents, in 10% of the murder mystery documents, and in 2% of the automotive documents. If this word occurs in a new document, the likelihood score will be multiplied by 0.12 for the sports label, 0.1 for the murder mystery label, and 0.02 for the automotive label. The overall effect will be to reduce the likelihood score of the murder mystery label slightly more than for the sports label, and to significantly reduce the score of the automotive label with respect to the

other two labels.

A likelihood score can be thought of as an estimate of the probability that a randomly selected document from the training set would have both the given label and the set of features, assuming that the feature probabilities are all independent. The naive independence assumption is unrealistic, as features are often highly dependent on one another. Despite this rather optimistic assumption, naive Bayes classifiers often outperform far more sophisticated alternatives. The reasons are related to Figure 2.6; although the individual label density estimates may be biased, this bias might not hurt the posterior probabilities as much, especially near the decision regions. In fact, the algorithm may be able to withstand considerable bias for the savings in variance that the “naive” assumption earns.

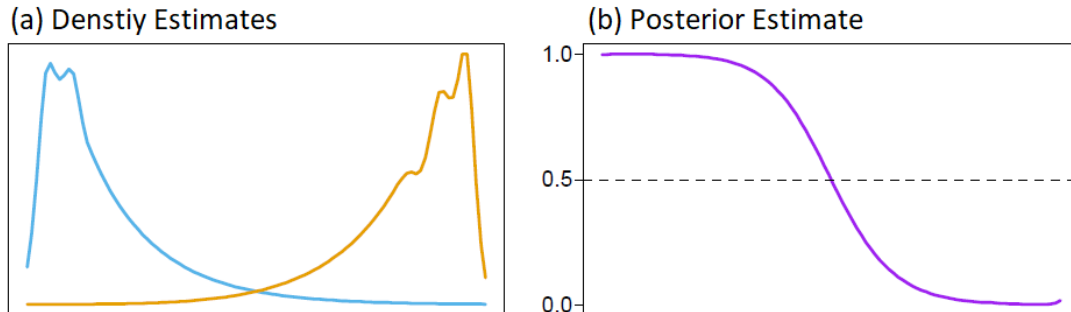


Figure 2.6: The density estimates of labels might have interesting structure (a), that disappears when the posterior probabilities are formed (b) (Hastie et al., 2009).

Naive Bayes classifiers have worked quite well in many real-world situations, for instance in topic classification and spam filtering. The naive independence assumption makes the algorithm easy to apply, and it requires less data to get a good result in many cases. However, because of the assumption, there are certain problems that Naive Bayes cannot solve. When it comes to tasks that have higher dependency between features, like most of the subtasks of IE, sequence classifiers are preferred.

2.6 Evaluation in Information Extraction

In order to decide whether a classification model is accurately capturing a pattern, the model must be evaluated. The result of this evaluation is important for deciding how trustworthy the model is, and for what purposes it can be used. In addition, the evaluation can be an effective tool for guiding the process of making future improvements to the model.

2.6.1 The Test Set

Most evaluation techniques in IE use an allocated test set consisting of labeled data to evaluate the classification model. Thus, the performance of the model can be evaluated by comparing the labels in the test set created by people, to the labels generated by the model. A high number of generated labels identical to the manually created labels indicates that the model is performing well, and classifies to expectations. It is vital that the corpus of the test set is different from that of the training set: if the training set is simply re-used for testing, the evaluation results would imply high performance, independent of the model's performance on new, unseen data.

When building the test set, there is often a trade-off between the amount of data available for testing and the amount available for training. The recommended distribution of data for training and testing varies from 60%/40% to 90%/10%. Generally, the training set should be larger than the test set to prioritize the "real" accuracy of the model, and the test set should be large enough to ensure a satisfactory minimum occurrence of all labels. This means that classification tasks with a large number of defined labels, or infrequent occurrences of labels, require larger test sets than classification tasks with fewer or more frequently occurring labels. Additionally, if the test set contains many closely related instances, such as instances drawn from a single document, then the size of the test set should be increased to ensure that this lack of diversity does not skew the evaluation results.

Optionally, a portion of the labeled documents can be set aside as a blind set. The point of using a blind set is to prevent the accuracy from being tainted by, for example, making changes based only on observed labels in processed documents. While the test set should be studied in detail to assist in iterative tuning of the model, the blind set should only be an indicator of the overall performance and not be used to influence the model.

2.6.2 Precision and Recall

To precisely evaluate classification models, a set of parameters are conventionally employed (Bird et al., 2009):

- **True Positives (TP)** are relevant items that were correctly identified as relevant.
- **True Negatives (TN)** are irrelevant items that were correctly identified as irrelevant.
- **False Positives (FP)**, or Type I errors, are irrelevant items that were incorrectly identified as relevant.
- **False Negatives (FN)**, or Type II errors, are relevant items that were incorrectly identified as irrelevant.

These parameters allow for calculation of precision (eq. 2.5) and recall (eq.2.6):

$$precision = \frac{TP}{(TP + FP)} \quad (2.5)$$

$$recall = \frac{TP}{TP + FN} \quad (2.6)$$

Precision indicates how many of the items that were identified were relevant, while recall indicates how many of the relevant items were identified. Thus, precision and recall can be seen as measures of correctness and completeness, respectively (Piskorski and Yangarber, 2013). The *F*-measure gives the harmonic mean of precision and recall, and the general formula is defined as:

$$F = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall} \quad (2.7)$$

In the general formula for the *F*-measure, β is a non-negative value used to adjust the relative weighting of precision and recall. A β -value of 1 gives equal weighting to precision and recall and is referred to as the "*F1* score". β -values lower than 1 give increased weight to precision, while β -values higher than 1 give increased weight to recall.

Chapter 3

Case: IE from Shipping Industry Emails

3.1 Motivation

Unstructured text can be mined for specific information that is unique to an organization's industry or business needs. NLP is a core function for parsing and identifying significant words in language, and IE is a subcategory of NLP concerned with obtaining useful information in a structured format. Being able to customize the IE system is essential to realize the full value of IE in domain specific processes.

A large scope of tasks in IE can be accomplished by the use of carefully constructed regular expressions (regexes). Examples of entities amenable to such extractions include email addresses, social security numbers, gene and protein names, etc. These entities share the characteristic that their key features are expressible in standard constructs of regular expressions. Manually created regexes remain a widely adopted practical solution for IE (Fukuda et al., 1998; Zhu et al., 2007). Similarly, manually created, or pre-existing lists (referred to as dictionaries) can be used to extract entities with an anticipated occurrence.

With manual, rule-based approaches to IE, like regexes and dictionaries, systems have been able to extract entities with relatively high accuracy for certain tasks. The task of extracting information from shipping industry emails, addressed in this case, has so far been handled with regexes. Based on feedback from shipbrokers using such systems, an accuracy of 85% is estimated for the systems' performance (P. Hafsaas, personal communication, May 11 2017). The potential for improved accuracy with the help of ML is evaluated.

A system for IE which makes use of ML is employed with the goal of assessing the practicality of the system in processes relevant to AVO Consulting.

3.2 Method

IBM Watson Knowledge Studio (WKS) is a cloud-based application that can be used to automatically extract information from text with the help of ML. A set of documents with annotated information of interest is needed for the supervised training of an ML model. Documents are annotated by people in the Ground Truth Editor (GTE). The tool is designed to be easy to use, so that the development of new systems relies on subject matter knowledge, rather than coding skills. Annotated documents can be used to train a custom ML model to automatically process the language of the domain.

WKS has linguistic analysis features incorporated in the software, making the system able to recognize basic rules of natural language, prior to fitting the ML model to a domain. The linguistic features include tokenization, POS tagging, parsing, sentence segmentation and semantic role labeling. These features make it possible to apply IE to new domains quickly and accurately, since the system only requires documents with annotations for entities, relationships and co-references of interest to train the ML model.

WKS currently supports nine languages; English, German, Japanese, Arabic, French, Korean, Spanish, Italian and Brazilian Portuguese. Norwegian is not supported, but several companies are showing interest in the support for this language (Hoemsnes, 2017; Staines, 2016). During the Workshop on WKS, Mark Rice stated that support for the Norwegian language would require development of a new lexical dictionary and a new POS tagging system, while the same algorithms used by other supported languages, using the Latin alphabet, can be used for tokenization and sentence segmentation (personal communication, March 29, 2017).

WKS has two environments; UIMA and SIRE. The tools of these environments, and the recommended practices for using them, are described on the basis of the WKS documentation (IBM, 2016).

3.2.1 Unstructured Information Management Architecture (UIMA)

A valuable option of WKS is the opportunity to use rule-based tools in combination with ML, thereby benefitting from the advantages of both techniques. In WKS, the environment for building rules is called Unstructured Information Management Architecture (UIMA) and consists of three rule-based tools; rules, dictionaries and regexes. These tools can be used to pre-annotate documents with entity mentions and potentially accelerate the work of manual annotation. The tools are refined in the UIMA interface, as shown in Figure 3.1.

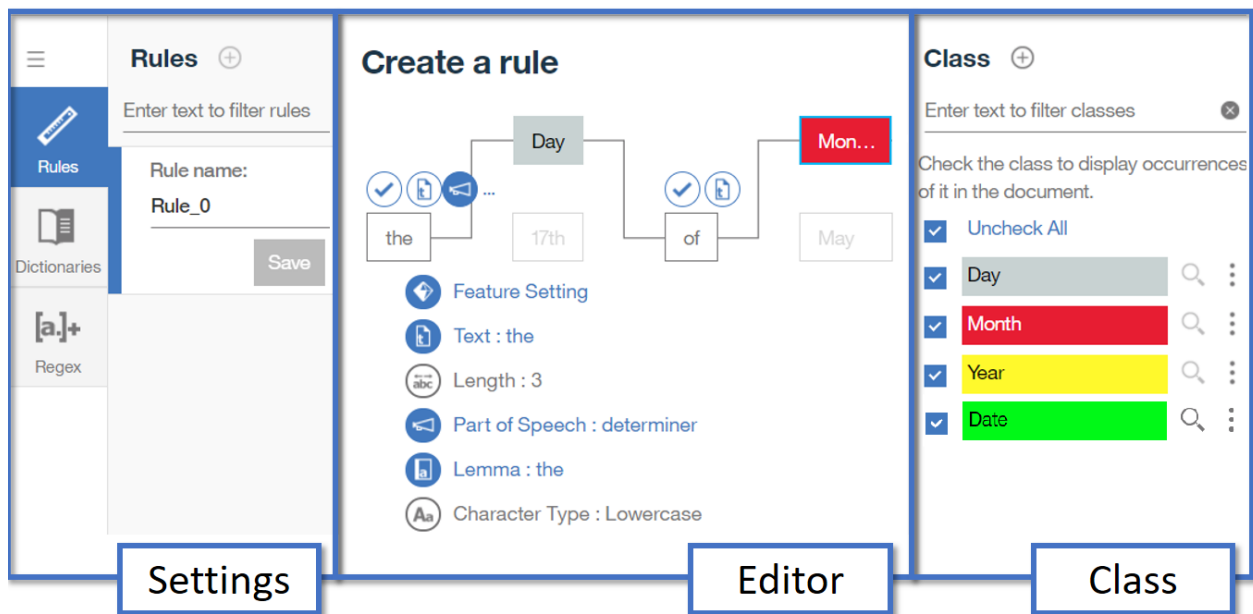


Figure 3.1: The UIMA interface. Rules, dictionaries and regexes can be created in the settings window. The tools' functions are refined in the editor window. Classes are defined in the class window.

During rule creation, classes are used to represent information. Classes are basically the same as entity types. The difference is that classes can have an intermediate form, which allows for more abstract annotations and the combination of several classes to define a more comprehensive class. For example, as shown in Figure 3.1, rules can be used to recognize a date. One way to do this is by creating three classes; Day, Month and Year, and then use these to define a fourth class, Date. Conditions of the rules may include the determiner "the" appearing in front of the class Day, capitalization of the class Month, and the class Year consisting of four numeric characters, etc. To use this rule annotator, the Date class must be mapped to a corresponding

entity.

The three tools available in UIMA are described:

- **Regexes** use a sequence of characters that defines a search pattern to find classes in the document. The Regex tool that is included in UIMA recognizes expressions that follow the "java.util.regex.Pattern" syntax. Classes are later mapped to entities, and the rule annotator can create annotations for entity mentions found in the document, based on the defined regexes.
- **Rules** can be used to capture patterns that occur in documents and convey information about underlying classes. Rules are different from regexes, since they offer the option of using POS tags in their conditions to regularly find classes. For example, a condition for a class, Action, may be that it has a verb POS tag. Classes are later mapped to entities, and the rule annotator can then create annotations for entity mentions found in the document, based on the defined rules.
- **Dictionaries** can be used to find classes in documents that match terms in the dictionary. A dictionary annotator associated with an entity type can be used to pre-annotate these mentions in documents. Dictionaries can also be used to influence the ML training heavily, as the algorithms have a strong disposition to label terms according to the dictionaries, but are not totally deterministic. Dictionaries are most effective when used as indicators for manual annotation.

3.2.2 Statistical Information and Relation Extraction (SIRE)

SIRE refers to the ML environment of WKS. The SIRE toolkit provides trainable NER, RE and CO components. The components are trained with data that has been manually annotated in the GTE. Manual annotation consists of identifying and tagging entities mentioned in documents, mentions that reference the same entity, and the relationship between entities. The tags used for entities and relationships are first defined in the type system.

Optionally, subtypes and roles can be used to enable more precise entities. By providing subtypes, the ML model can be trained to differentiate between different mentions of the same entity. Roles can be used to differentiate between the literal meaning of a mention, and the actual role it plays in the sentence. For example, in the phrase “the White House vetoed the bill” the White House can be labeled as an entity; FACILITY, with a role of ORGANIZATION. This entity is defined in Figure 3.2.

| Entity Type Name | Roles | Subtypes |
|------------------|----------------------|-----------------|
| FACILITY | Select a role | Enter a subtype |
| | FACILITY (This type) | COMMERCIAL |
| | ORGANIZATION | DOMESTIC |
| | | INDUSTRIAL |
| | | OTHER |

Figure 3.2: An entity type created in the type system. This entity has defined roles and subtypes.

The challenge of using subtypes and roles for more precise entities is that it requires more training data with examples of the use of these tags. Without sufficient examples of subtypes and roles in the training data, the ML model will not be able to distinguish between these types of entities.

By learning the patterns of the manual annotations, new annotations can automatically be added by the ML model to large sets of new documents. The trained components use maximum entropy models like MEMM to automatically add these annotations.

When documents have been annotated in the GTE, WKS automatically allocates documents in three sets of 70% testing data, 23% test data and 7% blind data. This ratio can be adjusted to fit a specific task. The performance of a trained ML model can be evaluated by looking at the results of the annotated test data and blind data. The statistics window in SIRE provides the precision, recall and F1 score of the ML annotator component, as well as the following metrics:

- **Percentage of total annotations** shows how many words were annotated with a given entity type or relation type out of the total number of words that were annotated as any entity type or relation type in the test document set. This value gives an indication of how prevalent one type of mention is, compared to the other types in the ground truth.
- **Percentage of corpus density (by number of words)** shows the number of words that were annotated with a given entity type or relation type out of the total number of words in the test document set. This value gives an indication of how prevalent mentions of this type are compared to all of the words in the domain documents.
- **Percentage of documents that contain the type** shows how many documents contain a given entity type or relation type. This value gives an indication of how well the documents represent the domain, and a low percentage for key entity types or relation types is a sign of under-representation of that particular type in the documents.

In addition to these metrics, SIRE provides the possibility of comparing annotations in the test data, generated by the ML model, to the annotations added to the test data in the GTE. This comparison is presented in a confusion matrix. False positives (FP) are entity mentions that were incorrectly annotated as another entity type, and are listed in the column of the incorrectly annotated entity type in the confusion matrix. False negatives (FN) are entity mentions that were missed by the ML model, and are listed in the column labeled "O" in the confusion matrix.

The following confusion matrix example shows results of an ML annotator run on documents that deal with traffic incidents. There are two defined entity types; MANUFACTURER and MODEL.

Table 3.1: Confusion Matrix Example

| Entity Types | MANUFACTURER | O | MODEL |
|---------------------|---------------------|----------|--------------|
| MANUFACTURER | 515 | 44 | 5 |
| O | 0 | 800 | 2 |
| MODEL | 0 | 21 | 377 |

In this example, the ML model correctly annotates 515 mentions of the MANUFACTURER entity, while 44 mentions are missed, and 5 mentions are incorrectly annotated as mentions of the MODEL type. 800 words are correctly left unannotated, while 2 words that should not be annotated are incorrectly annotated as the MODEL entity. 21 mentions of the MODEL entity are missed, while 377 mentions of this entity are correctly annotated.

3.2.3 Analysis of Statistics

The scores for precision, recall and F1 reaches their best value at 1 and worst value at 0. A low precision score indicates a need to improve annotation consistency. A low recall score indicates a need to add more training data. A low F1 score is an indication of both poor precision and poor recall. Low scores can occur for many different reasons that depend on the domain, type system complexity, appropriateness of training documents, human annotator skills, and other factors. A confusion matrix with entities with particularly many of mislabeled or missed mentions indicates commonly confused entities. The performance of the ML annotator must be tuned to address low scores and poor entities in the confusion matrix.

3.2.4 Improvement of ML Model Performance

The process of adapting NLP to a new domain with WKS consists of the following steps:

1. Create/Adapt type system.
2. Obtain representative documents.
3. Use UIMA to pre-annotate documents with rules, regexes and dictionaries.
4. Annotate documents manually.
5. Use SIRE to train and test an ML model.
6. Evaluate the test results.
7. Cycle back if necessary.

After evaluating the test results, various actions can be taken to improve the performance of the ML model. Improvements are made iteratively by creating new versions of the model, until a satisfactory version is attained.

The actions required to improve the model depend on the problem. Here are some commonly occurring problems and the recommended steps for improvement.

- Low recall is often an indication of an insufficient amount of training data. The recommended size of the training set is around 300.000 words to ensure frequent occurrences of types.
- Low precision is often an indication of inconsistent annotation. This can be resolved by creating clear and concise annotation guidelines that show how to annotate mentions properly under a given set of circumstances. In addition, by having multiple human annotators review an overlapping set of documents, commonly confused types can surface, and inconsistency in manual annotation can be reduced.
- Low percentage of documents that contain a certain type is often an indication of a corpus that does not fully represent the domain. In this case, the type system and the documents need to be investigated to ensure that the corpus contains relevant types. A recommended minimum number of 50 occurrences of mentions in the training set applies to all entity types.
- If the documents have references to concepts that are important in the domain but are not represented anywhere in the type system, types need to be added to the type system that capture the missing concepts or relationships. However, the types system should be limited to only the most fundamental types, as redundant types can be misused, or not used at all, and aggravate the performance of the ML model.

After following appropriate steps for improvement, the ML model can be retrained to produce an updated version. Optionally, the original version of the ML model can be saved, in case resources must be restored in future iterations, or if the results of the new version are worse, in which case the version may be reverted. The resources captured in the new version include the corpus, type system, ground truth and evaluation results, while the dictionaries are excluded because of complications in handling large or varying types of dictionaries.

3.3 Procedure

A free trial version of WKS is used for this demonstration. See Appendix B for documentation on registration, browser requirements, assembling a team and creating a project. The free trial version includes 5 GBs of storage, and the creation of up to 5 projects. Only one user is permitted access, prohibiting the collaboration of multiple domain experts, which is recommended for development of practical projects. The free trial does not include the option to export ML models, which prohibits the demonstration of employing the system, but the process of employment will be described in Section 4.2; Conclusions and Further Work. With the acknowledgement of these restrictions, this version of WKS is considered appropriate for the demonstration purposes of the case.

To demonstrate the practices of IBM WKS, a model is trained for analyzing language in the shipping domain. I have access to 56 emails received by a shipbroker. The emails are outdated, but fit the purpose of training the model. Documents to be imported in WKS must be either CSV files or TXT files in UTF-8 format. The original file format of the emails is MSG, and these files are converted to TXT files in UTF-8 format. The emails are read through, and unnecessary text is removed before converting the files. The text selected for processing in one of the emails is shown as an example in Figure 3.3. By curating the emails, it is ensured that each email accurately represents the language of the domain that is to be processed.

A document set of 30 emails is created for human annotation. As best practice, the first set for annotation should be relatively small to define annotation guidelines early and standardize the process. The first annotations will often be incorrect, as the users rarely know the data well enough to accurately define and use the first type system.

AH Arrow Handy <handy@arrowship.com>  0

CTM IO update

ARROW HANDY - Athens
www.arrowship.com
 75532524/PYD 17/01/2017 11:14:38

Baard // Aris

CTM update;

B/haul or period preferred but try others

MV ST PAUL = (TESS 58) = SAILED GANGAVARAM TODAY
 ETA RBCT 31 JAN/01 FEB

MV NAVIOS ORIANA = (IMABARI 61) = OPEN BEDI, 25/26 JANUARY

DtIs;

MV ST PAUL
 TESS 58, NK CLASS BUILT 2010
 MARSHALL ISLANDS FLAG
 DWT ABT 58,000 MT ON ABT 12.8M
 TPC:57.46
 GRT/NRT: 32,309/19,439
 LOA/BEAM: ABT 189.99MT / 32.26 MT
 5 H/H ABT 72600 M3M
 4 X 30 MT CRANES, MAX 12 M3 X 4 SETS GRABS
 SPEED/CONS: 14/14.5 KN LADEN /BALLAST 32MT IFO+0.15MT MDO
 IDLE: 3 MT IFO + 0.15 MT MDO
 WORKING (24 HRS): 5.5 MT IFO + 0.15 MT MDO
 ALL DETAILS ABOUT

M/V NAVIOS ORIANA
 2012 BLT SHIPYARD IMABARI
 61,442/13.01 MTSW
 LOA/BEAM 199.9/32.24
 77,600 GRAIN 5 HO/5 HA
 (1: 18.4 X 18.72 , 2-5: 23.2X 18..72)
 CRANES 4 X 30.5 MTS SWL + 4 GRABS 12 M3
 ABT 14.5 / ABT 14 L ON ABT 27 MT IFO (380)
 IPI: ABT 1.2 MTIFO + 1.2 MT (BOILER) + 0.3 MT MDO
 IPW: ABT 4.0 MTIFO + 1.2 MT (BOILER + 0.3 MT MDO
 ALL DETAILS ABOUT

Relevant Information

++

Arrow Shipping Hellas SA = Indian Ocean Desk
 Tel: +30 211 102 6000

Figure 3.3: Example of selected text of an email. On the basis of the information, it can be concluded that the motor vessel St Paul is estimated available between the 31. of January and the 1. of February from port RBCT, while the motor vessel Navios Oriana is available between the 25. and the 26. of January from port Bedi.

3.3.1 Defining the Type System

Features are created in the type system by defining the types of entities and relationships that can be annotated in the documents. Entities and relationships that are interesting to look for in the domain can be created from scratch or imported from type systems of similar domains. In this case, four entities are defined based on what is believed to be important information to the shipbroker. The entities are;

- VESSEL defines the name of the vessel.
- PORT defines the port for chartering.
- DATE defines the date chartering.
- CONTRACT defines the type of shipping contract.

Mentions in the documents can be classified as one of these four entities. Two subtypes are defined for the DATE entity. The subtypes are OPEN and ETA, and may be used to differentiate between mentions of confirmed dates for chartering and estimated dates for chartering. Two relation types are defined;

- AVAILABLE defines the relationships between PORT and DATE.
- LOCATION defines the relationship between VESSEL and PORT.

The relation types may be used to differentiate between different chartering proposals in emails that mention more than one. This is done by annotating the appropriate relationships between associated entities.

3.3.2 Rule Creation

The rule-based tools of UIMA are used to pre-annotate mentions of the DATE entity. These mentions may potentially be annotated entirely by relatively complex regexes, but rules and dictionaries are used in addition to regex for their demonstrative quality. First, a selection of nine documents are imported in the UIMA interface as a sample of the corpus to test the tools. Three classes are defined; DictMonth, RegexYear and RuleDate. The classes DictMonth and RegexYear are combined with the RuleDate class, which is later mapped to the DATE entity.

Dictionary for annotating months

A dictionary, Month, is set up containing the twelve months of the year. The various ways of writing the names and abbreviations of the months are added to the surface forms. The entries are written in lower case, which enables annotation of all occurrences, regardless of case. The POS tag is set to noun for every month. The dictionary can be seen in Figure 3.4. When the dictionary is associated with the DictMonth class, all the sample documents are automatically annotated for every mention of a month reflected in the dictionary.

The screenshot shows the UIMA dictionary interface for 'Month'. At the top, the title 'Month' is displayed with the subtitle 'Language: English | 12 entries'. To the right is a search bar with the placeholder text 'Enter text to filter entries'. Below the title are three buttons: 'Add Entry', 'Import', and 'Export'. The main content is a table with five columns: 'Lemma', 'Surface Forms', 'Part of Speech', and 'Action'. Each row represents a month, with a checkbox in the 'Lemma' column and 'Edit' and 'Delete' links in the 'Action' column.

| <input type="checkbox"/> | Lemma | Surface Forms | Part of Speech | Action |
|--------------------------|-----------|-----------------------------------|----------------|---|
| <input type="checkbox"/> | december | december, dec, dec. | Noun | Edit Delete |
| <input type="checkbox"/> | november | november, nov, nov. | Noun | Edit Delete |
| <input type="checkbox"/> | october | october, oct, oct. | Noun | Edit Delete |
| <input type="checkbox"/> | september | september, sep, sep., sept, sept. | Noun | Edit Delete |
| <input type="checkbox"/> | august | august, aug, aug. | Noun | Edit Delete |

Figure 3.4: An excerpt of a dictionary created in UIMA. There is one entry for each month of the year.

Regex for annotating years

A regex is defined as the following:

| Regular Expression | Minimum Word Tokens | Maximum Word Tokens | Action | |
|---------------------------|---------------------|---------------------|--------|--------|
| <code>(?:[0-9]{2})</code> | 1 | 1 | Update | Cancel |

Figure 3.5: A regex created in UIMA.

This regex finds numeric values in text which represent years between 1900 and 2099. The minimum and maximum word tokens are set to 1, meaning that the term can not consist of more than one token, such as other words, hyphens, etc.

Rule for annotating dates

Next, a rule can be defined for capturing a sequence of text that makes a date interval. This is done by selecting a text sequence from the sample documents that exemplifies this occurrence. Conditions of the rule are then defined, based on the characteristics of the sample sequence. The text sequence "5/9 September 2017" is selected from one of the sample documents. The conditions are defined as shown in Figure 3.6.

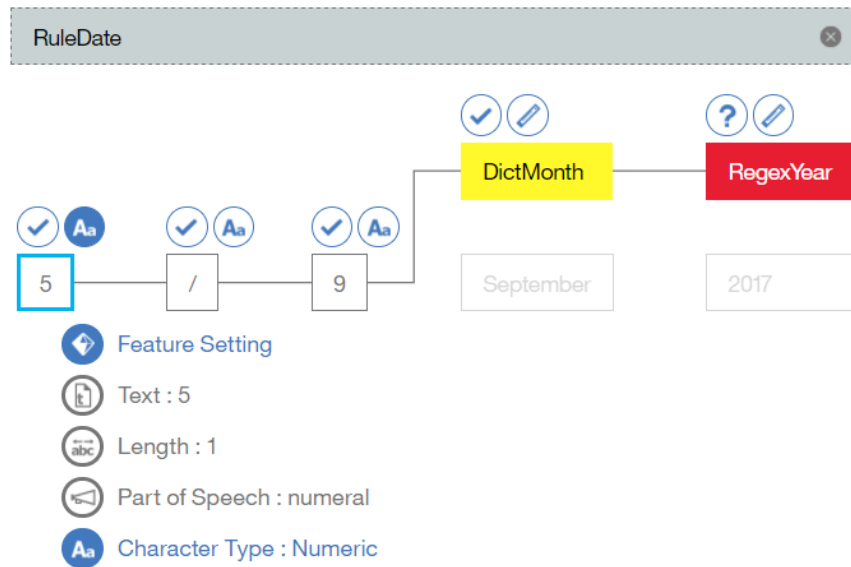


Figure 3.6: A rule created in UIMA with conditions for the first word in the sequence.

One condition of the rule is that the first word of the sequence consists of numeric character types. Similarly, the second word must be a punctuation mark, such as a slash. The third word must be a number, the fourth word must match class DictMonth, and the last word must match class RegexYear. The rule requires all the words to appear exactly one time, apart from the last word, RegexYear, which may appear zero or one time. The setup of this condition for RegexYear can be seen in Figure 3.7.

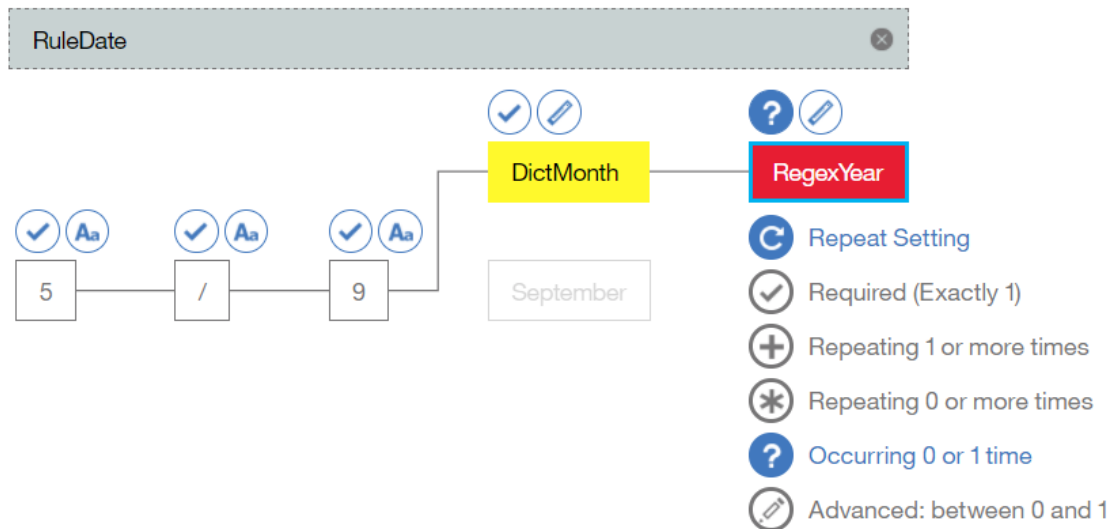


Figure 3.7: A rule created in UIMA with conditions for the last word in the sequence. The RegexYear class can occur once in the sequence, or not at all.

This rule is now able to capture the sequence "number + punctuation + number + DictMonth + (RegexYear)". A second rule is created to capture the sequence "number + DictMonth + punctuation + number + DictMonth". This sequence represents an alternative way of writing the date interval, which is observed in the sample documents. The class RuleDate is assigned to represent these sequences.

After saving the rules, they are automatically applied to the sample documents, and sequences in the documents that fulfill the conditions are annotated as the RuleDate class. In the sample document "CTM IO update.txt" the rules are able to correctly annotate "31 JAN/01 FEB" and "25/26 January" with the RuleDate class, as shown in Figure 3.8. The build year of one of the vessels is annotated by regex with the RegexYear class, but is not annotated by the rule component, since it does not fulfill its conditions.

CTM IO update.txt

B/haul or period preferred but try others
 MV ST PAUL = (Tess 58) = SAILED Gangavaram TODAY
 ETA RBCT 31 JAN /01 FEB
 MV NAVIOS ORIANA = (Imabari 61) = open Bedi, 25/26 January
 Dtls;
 MV ST PAUL
 TESS 58, NK CLASS BUILT 2010
 MARSHALL ISLANDS FLAG
 DWT ABT 58,000 MT ON ABT 12.8M

Figure 3.8: A document automatically annotated by the rule component.

The annotations generated by the UIMA tools in the sample documents can be seen for each individual class. The resulting annotations are displayed in Figure 3.9.

Find Annotations

Select a class from the Class panel or enter the class name

DictMonth RegexYear RuleDate Enter class name

Results: DictMonth 15 annotations RegexYear 12 annotations RuleDate 8 annotations

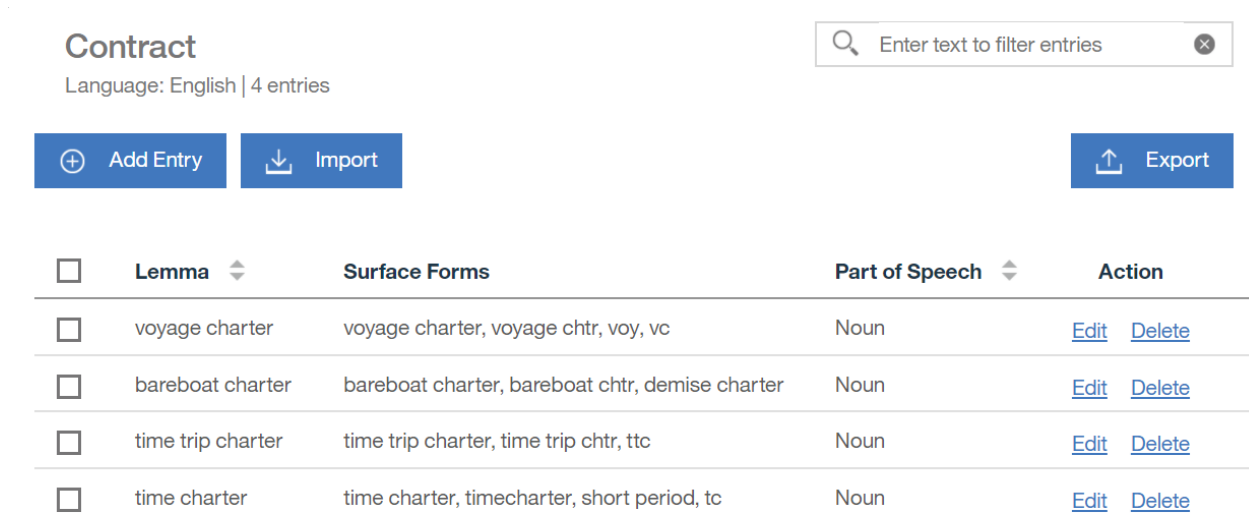
| Text | Document |
|---|---------------------|
| ... OPEN TAMPA 23 - 26 JANUARY 2017 ... | MV MARIA - DWT .txt |
| ... OPEN TAMPA 23 - 26 JANUARY 2017 ... | MV MARIA - DWT .txt |
| ... OPEN TAMPA 23 - 26 JANUARY 2017 ... | MV MARIA - DWT .txt |

Figure 3.9: Annotations generated by the rule, regex and dictionary in the sample documents.

In the nine sample documents fifteen terms are annotated by the dictionary, twelve terms are annotated as years by the regex and eight sequences are annotated as dates by the rule.

Dictionary for annotating contracts

A new dictionary is created (see Figure 3.10) by manually adding shipping contract types and assigning the dictionary to the CONTRACT entity. Four types of contracts are added with their corresponding synonyms and abbreviations as surface forms. Like the Month dictionary, entries are added in lowercase to include all occurrences, regardless of case, and the POS tags are set to noun.



Contract
Language: English | 4 entries

Enter text to filter entries

+ Add Entry ↓ Import ↑ Export

| <input type="checkbox"/> | Lemma | Surface Forms | Part of Speech | Action |
|--------------------------|-------------------|---|----------------|---|
| <input type="checkbox"/> | voyage charter | voyage charter, voyage chtr, voy, vc | Noun | Edit Delete |
| <input type="checkbox"/> | bareboat charter | bareboat charter, bareboat chtr, demise charter | Noun | Edit Delete |
| <input type="checkbox"/> | time trip charter | time trip charter, time trip chtr, ttc | Noun | Edit Delete |
| <input type="checkbox"/> | time charter | time charter, timecharter, short period, tc | Noun | Edit Delete |

Figure 3.10: A dictionary created in UIMA. The dictionary can be used to annotate the relevant shipping contract types.

The dictionary is only able to annotate one CONTRACT mention in the nine sample documents, which may be a sign that the entries in the dictionary do not occur frequently.

Dictionaries for annotating ports and vessels

Another way of utilizing the dictionary function of UIMA in this case, is to use a dictionary annotator to automatically annotate all the mentions of ports and vessels. This is done by gathering the names of all the operational ports and vessels and importing them in two separate dictionaries. These dictionaries are then associated with the PORT entity and the VESSEL entity, respectively. A list of 9406 port names and a list of 2500 vessel names were extracted from Marinetraffic (2007). The dictionary annotators are used to pre-annotate the first document set. The resulting documents have many words incorrectly annotated as the PORT and VESSEL entities. The

reason for this is that the dictionaries contain entries which can be used in several contexts, in addition to being a name of a port or a vessel. Therefore, these dictionaries are rejected and not used to pre-annotate the documents. As a general rule, dictionaries should not contain entries which can have multiple meanings.

3.3.3 Annotating in the Ground Truth Editor

The documents are manually annotated in the GTE. The color of annotations and optional keyboard shortcuts can be set up in the GTE settings. The task of annotating entity mentions is performed by selecting a string of text in a document and applying one of the entity types defined in the type system that most appropriately describes what that string of text represents. Some of the text has already been annotated by the RuleDate annotator and the DictContract annotator, and the remaining mentions of mentions are manually annotated. The DATE entity is specified as either the OPEN subtype or the ETA subtype. The procedure of annotating entity mentions is shown in Figure 3.11.

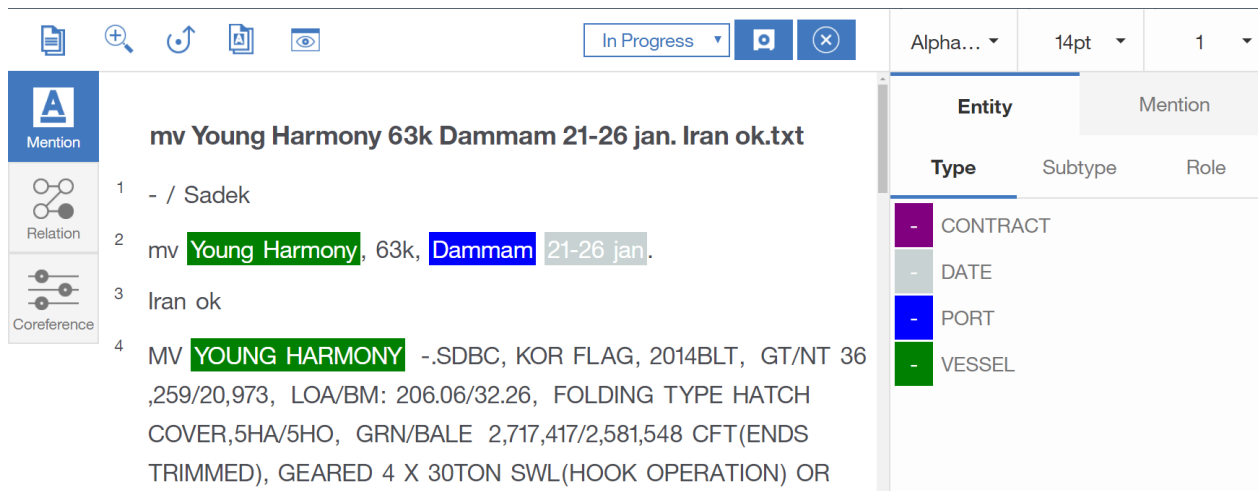


Figure 3.11: The entity mode in GTE. "Young Harmony" is annotated as a VESSEL entity, "Damman" as a PORT entity, and "21-26 jan" as a DATE entity, with subtype OPEN.

Relationships between mentions are annotated by connecting two entity mentions with a relation type defined in the type system. Relations which can be annotated in WKS are restricted to be between entity mentions within a single sentence. Annotation of relation types is done in the relation mode of GTE, shown in Figure 3.12.

Figure 3.12: The relation mode in GTE. A relationship between PORT and VESSEL is annotated LOCATION, and a relationship between PORT and DATE is annotated AVAILABLE.

Mentions that coreference the same entity are marked as coreferences. This helps the ML model recognize that entities which are referred to in different ways are to be associated with the same entity, thus helping to ensure consistency in the annotations where entity mentions are not identical. This is done in the coreference mode of GTE, shown in Figure 3.13.

Figure 3.13: The coreference mode in GTE. The two mentions "Young Harmony" refer to the same entity, and are therefore marked as a coreference chain.

Some annotation choices are unclear, like whether the whole text, "MV Young Harmony", or just the name, "Young Harmony", should be annotated as the VESSEL entity. The choice is made to annotate just the name, and the decision is kept throughout the annotation process.

Consistent annotation is at least as important as correct annotation when it comes to training an ML model. In addition, shorter entity mentions are better for training because shorter patterns are more easily recognizable by an ML model.

All the documents are read through carefully to ensure that no annotations are missed. In addition to learning from what is annotated, an ML model learns from what is not annotated. Thus, an entity mention that was not captured teaches the model to ignore that particular mention. This can potentially reverse gains made by annotating documents and lead to inaccuracy in the annotations generated by the ML model.

Training the ML model

Documents that have been annotated thoroughly are marked as complete. Once all the documents have been completed, the status of the annotation set changes from "in progress" to "submitted". The submitted documents are then evaluated, typically by a project manager, and either rejected or accepted. An accepted annotation set is promoted to the ground truth (previously referred to as the gold standard), which then can be used to train an ML model.

The percentage of documents used for training data, test data and blind data is set to the default value of 70/23/7. As a result, the first set consisting of 30 documents was partitioned in 21 documents for training data, 6 documents for test data and 3 documents for blind data. The process of training and evaluating the ML model is started and takes around ten minutes to complete in this case. The time it takes for the system to train and evaluate the model largely depends on the amount of training data used.

3.4 Results

The annotations generated in the test set are reviewed to determine whether any adjustments must be made to the ML model to improve its ability to find valid entity mentions, relation mentions, and coreferences in the documents.

3.4.1 Statistics of ML Model 1.0

A summary of statistics for entity types, relation types, and coreference chains can be viewed in the details page of the ML model. Figures 3.14, 3.15, 3.16, and 3.17 show statistics of the annotations made to the 6 test documents of the first set.


| Entity Types | F1 | Precision | Recall |
|--|-------------|-------------|-------------|
|  CONTRACT | 0 | 0 | 0 |
| DATE | 0.67 | 1 | 0.5 |
| PORT | 0.73 | 1 | 0.57 |
| VESSEL | 0.75 | 0.86 | 0.67 |
| Overall Statistics | 0.67 | 0.87 | 0.54 |

Figure 3.14: The F1 score, precision and recall for the entity types in the test set.

The CONTRACT entity is flagged and highlighted because it has an F1 value lower than the fixed value of 0.5. This indicates that the entity requires investigation and improvement. As seen in Figure 3.15, the CONTRACT entity rarely occurs in the test set. Only two out of the six test documents contain this entity, and it only makes up 5% of all the annotations. The values for the F1 score, precision and recall are all zero for the entity. The entity has far fewer mentions than the other entities, which leads to inaccurate annotations made by the ML model. The assumable reason for the low number of CONTRACT entity mentions is that most of the emails imply a time charter contract, without the need to specify the type of contract.

The recall of the DATE entity is 0.50, as seen in Figure 3.14. This signifies a relatively low amount of training data. This score is lower than for the PORT and VESSEL entities, despite the fact that DATE makes up for the second most annotations out of all the entities, with 34% of the total annotations. The low recall may be caused by the subtypes defined for the DATE entity. The subtypes make the entity more precise, by allowing differentiation between the estimated dates (ETA) and the confirmed dates (OPEN) for chartering. However, to accurately recognize this difference, the ML model requires more training data with examples of the use of these subtypes.






| Entity Types  | % of Total Annotations  | % of Corpus Density (by number of words)  | % of Documents that Contain This Type  |
|--|--|--|---|
|  CONTRACT | 5% (2/41) | 0% (2/1061) | 33% (2/6) |
| DATE | 34% (14/41) | 1% (14/1061) | 83% (5/6) |
| PORT | 20% (8/41) | 1% (8/1061) | 100% (6/6) |
| VESSEL | 41% (17/41) | 2% (17/1061) | 83% (5/6) |
| Overall Statistics | 100% (41/41) | 4% (41/1061) | 100% (6/6) |

Figure 3.15: The percentages for the entity types in the test set.

An observation made from Figure 3.15 is that the percentage of corpus density is very low for all the entities. Only 4% of the words in the test documents are mentions of one of the four entities. This indicates that there's a low representation of entity mentions compared to the amount of text in the documents, meaning that the corpus does not represent the language handled in this case very well. However, the corpus density is similarly low across all entities, meaning that they are equally prevalent. The exception is the CONTRACT entity, which has the lowest frequency of mentions by far, with only two mentions in the ground truth of the test set.

Figure 3.16 shows that the F1 scores for relation types AVAILABLE and LOCATION are relatively low. This is not due to consistency of annotations, as seen by the perfect precision values, but rather insufficient training data, as seen by the low recall values. The precision is presumably high because the relation types only can be annotated between two entities defined in the type system; VESSEL and PORT for LOCATION, and PORT and DATE for AVAILABLE. The recall, on the other hand, is most likely low due to the rare occurrence of these relation types in the test set.

| Relation Types | F1 | Precision | Recall |
|---------------------------|-------------|-----------|-------------|
| AVAILABLE | 0.5 | 1 | 0.33 |
| LOCATION | 0.67 | 1 | 0.5 |
| Overall Statistics | 0.55 | 1 | 0.38 |

Figure 3.16: The F1 score, precision and recall for the relation types in the test set.

Figure 3.17 only shows the statistics of the VESSEL coreference chains. This is because the VESSEL entity was the only entity type which was mentioned several times in emails in the test set. The F1 score for the VESSEL coreference chain is relatively high, presumably because most of the coreferenced mentions have the exact same syntax, and is therefore easily recognizable by the ML model.

| Entity Types | F1 | Precision | Recall |
|---------------------------|------------|-----------|-------------|
| CONTRACT | N/A | N/A | N/A |
| DATE | N/A | N/A | N/A |
| PORT | N/A | N/A | N/A |
| VESSEL | 0.8 | 1 | 0.67 |
| Overall Statistics | 0.8 | 1 | 0.67 |

Figure 3.17: The F1 score, precision and recall for the coreference chains in the test set.

The statistics show high precision for all entity types, relation types and coreference chains. This indicates high annotation consistency, meaning that the ML model rarely annotates irrelevant mentions, relations or coreferences. This tendency can also be observed in the confusion matrix shown in Figure 3.18. The rows of the matrix represent annotations from the ground truth, while the columns represent annotations made by the ML model. "O" indicates words that were not annotated.

| Entity Types | CONTRACT | DATE | O | PORT | VESSEL | Total |
|--------------|----------|----------|-------------|----------|-----------|-------------|
| CONTRACT | 0 | 0 | 2 | 0 | 0 | 2 |
| DATE | 0 | 6 | 8 | 0 | 0 | 14 |
| O | 1 | 0 | 1019 | 0 | 0 | 1020 |
| PORT | 0 | 0 | 4 | 4 | 0 | 8 |
| VESSEL | 0 | 0 | 5 | 0 | 12 | 17 |
| Total | 1 | 6 | 1038 | 4 | 12 | 1061 |

Figure 3.18: The confusion matrix for the entity types in the test set.

The confusion matrix shows that there are two annotations for the CONTRACT entity in the ground truth which the ML model fails to annotate. The ML model incorrectly annotates a word as a CONTRACT entity mention that should not have been annotated. The PORT and DATE entities have about half correctly annotated mentions and half missed annotations, while the VESSEL entity has five missed annotations out of seventeen. Words that should not be annotated are by far the most common in the test documents, with 1020 words out of 1061 total words, which might influence the ML model negatively. The relatively high numbers of missed annotations in the confusion matrix suggest that the ML model has a disposition to not annotate words.

3.4.2 Annotating with the ML Model

Once an ML model has been produced, it can be applied for annotation of new documents. A second document set is created from the remaining 26 emails and is annotated by the ML model. This reduces the efforts of manually annotating the remaining documents. In addition, it makes it possible to evaluate the initial performance of the ML model by directly seeing the annotations it makes. By observing annotations made to new documents, potentially overfit ML models can be discovered and adjusted by retraining with different training data. The annotations made can be seen in the GTE of the second set once the ML model has been applied for annotation. Figures 3.19, 3.20, and 3.21 show the annotations that the ML model was able to generate in the second document set.

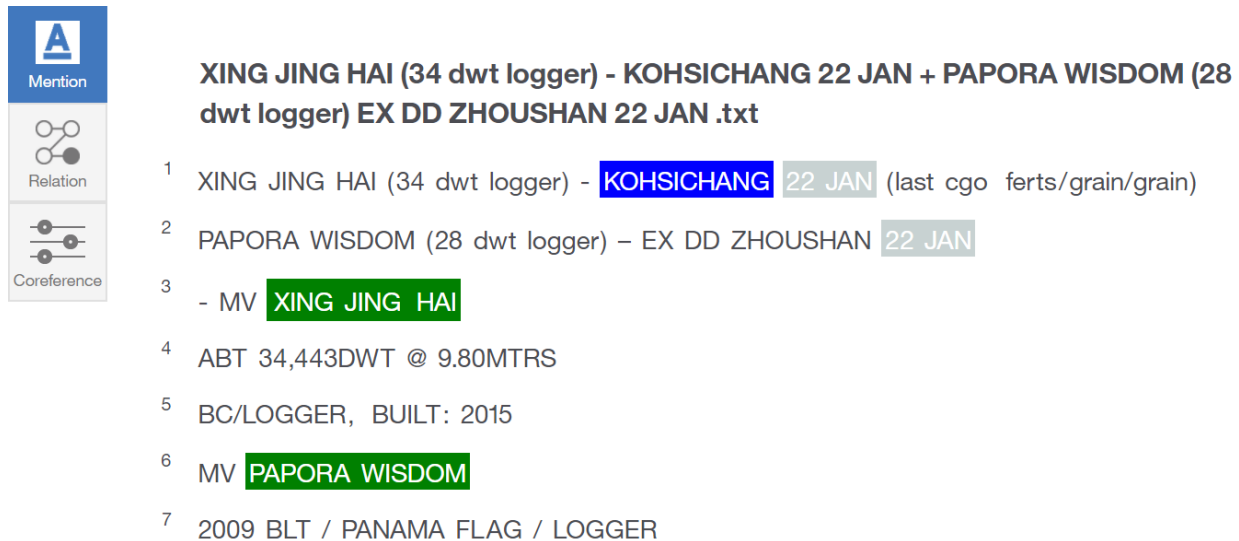
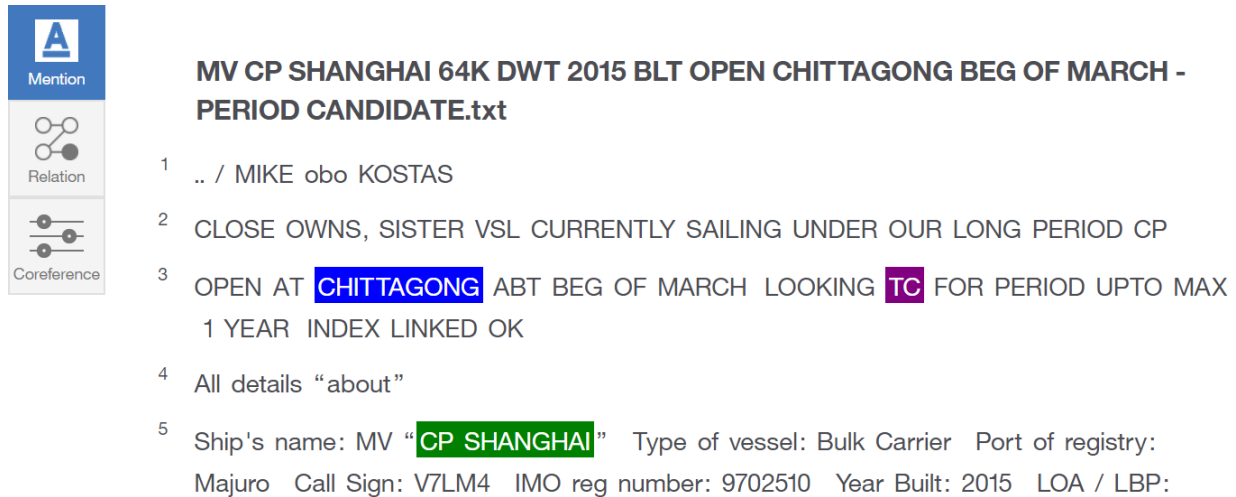


Figure 3.19: Annotation of entity mentions generated by the ML model to one of the documents in the second set.

The VESSEL entity mentions; "XING JING HAI" and "PAPORA WISDOM" were first missed and then annotated in later occurrences. This may be due to the ML model expecting the abbreviation "MV" in front of the entity, which does not appear in the first occurrences of the VESSEL mentions. This can be a sign of an overfit model that has learned a pattern in the training data to the degree that it does not recognize exceptions in new data. The PORT entity mention; "ZHOUSHAN" is missed by the ML model, possibly because of unexpected abbreviations appearing in this instance. These mistakes can be resolved by adding more documents with varying occurrences of mentions to the training set.



MV CP SHANGHAI 64K DWT 2015 BLT OPEN CHITTAGONG BEG OF MARCH - PERIOD CANDIDATE.txt

1 .. / MIKE obo KOSTAS

2 CLOSE OWNS, SISTER VSL CURRENTLY SAILING UNDER OUR LONG PERIOD CP

3 OPEN AT **CHITTAGONG** ABT BEG OF MARCH LOOKING **TC** FOR PERIOD UPTO MAX 1 YEAR INDEX LINKED OK

4 All details "about"

5 Ship's name: MV "**CP SHANGHAI**" Type of vessel: Bulk Carrier Port of registry: Majuro Call Sign: V7LM4 IMO reg number: 9702510 Year Built: 2015 LOA / LBP:

Figure 3.20: Annotation of entity mentions generated by the ML model to one of the documents in the second set.

The DATE entity mention; "BEG OF MARCH" is missed by the ML model, presumably because it rarely, or never, occurs in this format in the training set. This mistake may be avoided by adding conditions for this particular syntax to the RuleDate, or by adding more documents to the training data that contain this type of syntax.

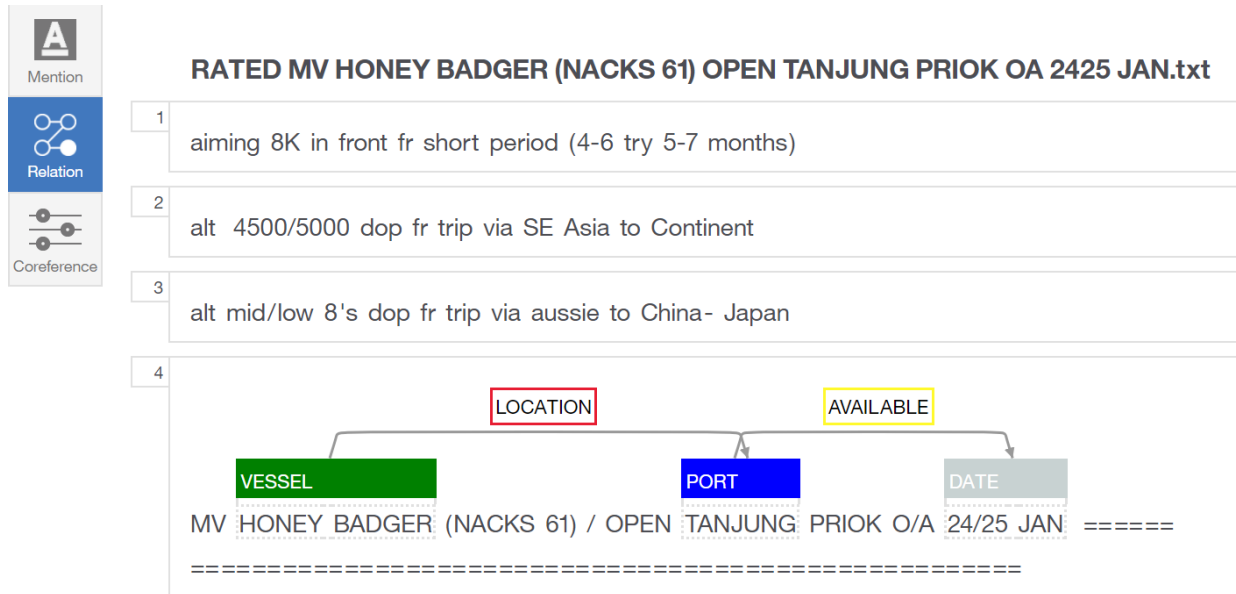


Figure 3.21: Annotations of entity mentions and relation mentions generated by the ML model to one of the documents in the second set.

The entity types; VESSEL, PORT and DATE, as well as the relation types; LOCATION and AVAILABLE are correctly annotated by the ML model. However, the CONTRACT entity mention "short period" is missed by the ML model. This is presumably caused by the infrequent occurrences of this entity in the training documents.

3.4.3 Retraining the ML Model

Some adjustments are made to the type system and the new documents based on the statistics of the ML model and the observed annotations it makes.

The low percentage of corpus density across all entities indicates a problem with the representative properties of the corpus. Extensive curation of the documents is conducted in an attempt to improve the corpus. Parts of the emails which do not contain relevant entity types, such as some of the vessel details, are removed from the documents of the second set before re-importing them in WKS.

The CONTRACT entity rarely occurs in emails. The mentions of this type are considered too infrequent, and the entity therefore rejected from the type system. The subtypes of the DATE entity are also removed, since the available training data is limited and considered to be inadequate for training precise subtypes.

After adjustments are made to the type system, the updated type system is applied to the GTE of both document sets. This automatically removes the annotations of the CONTRACT entity and the subtypes of the DATE entity from the first set. The second set requires manual annotation of the remaining entity mentions, relations and coreferences that were not annotated by the ML model.

When annotations for the second set are complete, the set is submitted and the ML model is retrained with all the 56 annotated documents. This is done by creating a new distribution of documents in the model settings window, seen in Figure 3.22.

HOME / Case / Annotator Component / Machine Learning Model

Versions Statistics **Model Settings**

Train Evaluate Train & Evaluate Go Back to View Mode

Training / Test / Blind Sets Dictionary Mapping

| Document Set | Task Status |
|--|-------------|
| <input type="checkbox"/> All | |
| <input checked="" type="checkbox"/> SET1 | COMPLETED |
| <input checked="" type="checkbox"/> SET2 | COMPLETED |

Create new sets by splitting the selected document...

Ratio
Enter the percentage of documents to include in each set.

Training Set (70% Recommended)

Test Set (23% Recommended)

Blind Set (7% Recommended)

Add documents in the selected sets to:

| Set Name | Number of Documents |
|--|---------------------|
| <input checked="" type="checkbox"/> Training Set | 42 |
| <input type="checkbox"/> Test Set | 14 |
| <input type="checkbox"/> Blind Set | |

Figure 3.22: The model settings of the ML model.

The documents are distributed in sets of 75% training data, 25% test data, and 0% blind data, resulting in a training set of 42 documents and a test set of 14 documents. The blind set is migrated to the other two sets, since all of the annotations made to the second set have been observed in the GTE, making the blind set less unseen and thereby less purposeful. A new version of the ML model is then trained and evaluated.

3.4.4 Statistics of ML Model 1.1

The performance of version 1.1 of the ML model is analyzed by reviewing the summary of statistics for the model. Figures 3.23, 3.24, 3.25 and 3.26 show the statistics for entity types, relation types and coreference chains in the new test set.

| Entity Types | F1 | Precision | Recall |
|---------------------------|------------|-------------|-------------|
| DATE | 0.9 | 0.93 | 0.87 |
| PORT | 0.8 | 0.86 | 0.75 |
| VESSEL | 0.69 | 0.83 | 0.59 |
| Overall Statistics | 0.8 | 0.88 | 0.73 |

Figure 3.23: The F1 score, precision and recall for the entity types in the new test set.

The overall statistics for the entities in the new document set have improved with the added training data and the adjustments made to the type system and the corpus. The recall values of all the entities are still lower than the precision values. This impacts the F1 scores, or the weighted averages, negatively. Low recall values mean that the ML model is missing some mentions altogether, and is typically a result of insufficient training data. The 56 imported documents have a total size of about 6.000 words, which is much smaller than the recommended size of 300.000 words. This indicates that the results may be improved significantly with more added training data.

The precision and recall for the VESSEL entity is lower than in the original test set. The annotations of this entity need to be reviewed in the confusion matrix and in the decoding results.

| Entity Types | % of Total Annotations | % of Corpus Density (by number of words) | % of Documents that Contain This Type |
|---------------------------|------------------------|--|---------------------------------------|
| DATE | 40% (32/81) | 1% (32/2484) | 93% (13/14) |
| PORT | 22% (18/81) | 1% (18/2484) | 100% (14/14) |
| VESSEL | 38% (31/81) | 1% (31/2484) | 100% (14/14) |
| Overall Statistics | 100% (81/81) | 3% (81/2484) | 100% (14/14) |

Figure 3.24: The percentages for the entity types in the new test set.

The corpus density is still alarmingly low for all the entities. This is due to the contents of the first document set, which is not extensively curated like the second set due to the limited time available.

| Relation Types | F1 | Precision | Recall |
|---------------------------|-------------|-------------|-------------|
| AVAILABLE | 0.75 | 0.75 | 0.75 |
| LOCATION | 0.4 | 1 | 0.25 |
| Overall Statistics | 0.69 | 0.77 | 0.63 |

Figure 3.25: The F1 score, precision and recall for the relation types in the new test set.

The relation type LOCATION in Figure 3.25 is highlighted because of its low F1 score, which indicates that the entity requires investigation and improvement.

| Entity Types | F1 | Precision | Recall |
|---------------------------|------------|-----------|-------------|
| DATE | N/A | N/A | N/A |
| PORT | N/A | N/A | N/A |
| VESSEL | 0.5 | 1 | 0.33 |
| Overall Statistics | 0.5 | 1 | 0.33 |

Figure 3.26: The F1 score, precision and recall for the coreference chains in the new test set.

The statistics for coreference chains in Figure 3.26 show that the VESSEL entity still is the only entity type that is coreferenced in the test set. The recall is lower than in the original test set, which is a development of the low recall value of the actual entity. More missed entity mentions of the VESSEL entity leads to less mentions available for coreferencing. Next, the statistics presented in the confusion matrix in Figure 3.27 are analyzed.

| Entity Types | DATE | O | PORT | VESSEL | Total |
|--------------|-----------|-------------|-----------|-----------|-------------|
| DATE | 27 | 2 | 0 | 0 | 29 |
| O | 2 | 2262 | 2 | 2 | 2268 |
| PORT | 0 | 4 | 13 | 0 | 17 |
| VESSEL | 0 | 8 | 0 | 21 | 29 |
| Total | 29 | 2276 | 15 | 23 | 2343 |

Figure 3.27: The confusion matrix for the entity types in the new test set.

As seen in the confusion matrix, version 1.1 of the ML model generates fewer erroneous annotations and misses fewer entity mentions than version 1.0, relative to the number of annotations. The VESSEL entity is correctly annotated by the ML model 26 times, while 8 mentions are missed and 2 words which should not be annotated are mislabeled as the VESSEL entity. The VESSEL entity mentions are missed the most by the ML model. This indicates that there is an inaccuracy related to the annotation of this entity.

3.4.5 Decoding Results

The decoding results show the annotations generated by the ML model in the new test set. Figures 3.28, 3.29 and 3.30 display the decoding results in three different test documents.

MARATHA PROMISE ZHOUSHAN 20 JAN.txt

- 1 / Oliver
- 2 After we clean fixed Maratha Prestige for period the below remain
- 3 Holding low 5k for first 25ds and 6k thereafter for period
- 4 Pref flat rate at low 6k
- 5 Any interest for period?
- 6 MARATHA PROMISE (37 dwt) – ZHOUSHAN 20 JAN
- 7 M. V. MARATHA PROMISE
- 8 BUILT / FLAG : 2012, ONOMICHI, SAIKI YARD / MARSHALL ISLANDS

Figure 3.28: Decoding results of one of the test documents. The ML model misses mentions of the VESSEL entity "MARATHA PROMISE".

MV AGONISTIS .txt

- 1 Baard // Aris
- 2 Remain – EX OUR PERIOD CP
- 3 = vsl with clse head owners
- 4 = considering any
- 5 = prefer to avoid dirties
- 6 = last cgo: mang.ore
- 7 ETA RICHARDS BAY 31 JAN/3 FEB
- 8 (SAILED HALDIA 15 JAN)
- 9 MV AGONISTIS GRK FLG BLT 2010 58399 DWTAT ON 13.02 M LOA/BEAM 196/
32.26 M 75530 CBM GRAIN 5/5 HOHA 4 X 36 CRANES + GRABS 12.5/13 ON 28
NDAS

Figure 3.29: Decoding results of one of the test documents. The ML model incorrectly annotates the text "15 JAN" as a DATE entity. The text "AGONISTIS GRK" is incorrectly annotated as a VESSEL entity, which should consist of just the text "AGONISTIS".

The decoding results in Figures 3.28 and 3.29 show that the ML model has issues when it comes to annotating the VESSEL entity. This might be caused by the irregular appearance of the entity. There are many variations of vessel names and the ML model has trouble recognizing them. In some cases however, such as the one shown in Figure 3.30, the ML model is able to correctly annotate the whole document.

MITSUI 56 MV LEO ADVANCE - KUSHIRO JAPAN ON 21ST JAN.txt

1 Pls pps:
 2 - MITSUI 56 MV **LEO ADVANCE** opening in **KUSHIRO**, JAPAN on **21ST JAN**
 3 (ROB HFO abt 950-1050 MT, LSGO abt 150-170MT)
 4 =====
 5 NAME : **LEO ADVANCE**
 6 FLAG : PANAMA
 7 BUILT : 2007, MITSUI TAMANO

Figure 3.30: Decoding results of one of the test documents. The ML mode correctly annotates all entity types, relation types and coreference chains, according to the ground truth.

Chapter 4

Summary

This thesis presents the theoretical framework of utilizing ML in NLP, with a focus on the task of extracting information. The theory chapter first addresses unstructured data, and why such data is difficult to process. An overview of the available methods for training an ML model is provided. The thesis concentrates on supervised learning, since this is the most commonly used training method in practical systems. Some of the algorithms and evaluation methods commonly used in IE are described with the goal of providing insight in the functionality of current IE systems.

Furthermore, the thesis gives a proposal of design of an example application, using IE technology available to AVO Consulting. The proposed application is implemented in WKS. The implementation procedure is described with focus on best practices recommended in the discussed theory. Complete implementation of the application requires deployment of the trained ML model. This is not achieved because of the limitations of the employed version of WKS.

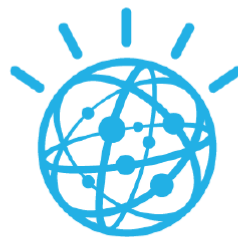
Available data is used for training two versions of the ML model, and the quality of the obtained application is assessed by comparing the results of the two versions. The results of the second version indicate improvement in the performance of the ML model. The statistics of the results of the second set give an overall F1 score of 0.8, while the overall F1 score of the first version is 0.67. The measures that were carried out for improvement include adding more training data, curation of documents, and removal of underperforming types and subtypes in the type system.

4.1 Discussion

IE is an emerging technology with the potential to transform the way textual information is analyzed and used for many information analysis tasks. When an application has access to the ML model of an IE system, unstructured data can be passed to the model to automatically extract relevant information in the structured form of entity types and relation types. This structured data can be used for several purposes, depending on the goals of specific tasks. For the shipping case, a possible goal may be to automatically create a schedule of shipping plans, with accompanying port names and vessel names to decrease the analytic time and improve the accuracy of the chartering process. In general, the goal of automation is typically to achieve faster and more accurate decision making. With IE, information from unstructured data can be made readily available for use in the decision making process. Figure 4.1 shows an example of WKS being applied in a customer complaint case. In customer-centric processes like this example, unstructured data plays an important role for decision making.

Unstructured data

Commander 4.0 Cu. Ft.
26-Cycle King-Size washer – white.
I hate this machine. Have had 3 calls on machine. You can't wash large items, Won't clean in the middle. Leaves dry spots through the clothes, I can only do ½ basket of clothes. Will not clean or mix bleach in with the water...



Natural-language
analysis (WKS)

Structured data for analysis

| | |
|----------|--------------------|
| Product | Commander |
| Category | 4.0 Cu. Ft. |
| Size | 26-Cycle King-Size |
| Model | washer |
| Color | white |
| Issue | large items |
| Issue | leaves dry spots |
| Issue | ½ basket |
| Issue | not clean |
| Issue | mix bleach |
| Problem | Insufficient water |

Figure 4.1: The unstructured content of a customer complaint and the corresponding structured form generated by WKS (Rice, 2017). The structured data is analyzed to identify the cause of the issues in the complaint. The knowledge acquired from analyzing structured data may be crucial in further processing.

Processes that involve analysis of unstructured data, like the one in Figure 4.1, are examples of higher-lever tasks, which can not be automated with the use of RPA alone. To fully automate these types of tasks, an IE system like WKS may be used in combination with RPA. Thus, the RPA software can be given access to information in unstructured data, and execute predefined decisions based on the information acquired.

IBM Watson Knowledge Studio was used to produce an ML model for annotation of shipping emails. WKS is an accessible platform for experimentation with data, and the free trial version provides features sufficient for creating a demonstrative system. The tools provided for training the model do not require coding skills. This is an advantage for the demonstration purposes of the case, as the procedures are more visually representable than code. This is also an advantage in practical systems, as it enables subject matter experts to work in the training process, thereby aggregating knowledge in the process. Knowledge of the corpus plays a vital part in domain specific projects.

The set of emails used in the case was chosen based on the availability of data in a language supported by WKS, and on the relevance to AVO Consulting's work. Ideally, language processed by WKS should incorporate grammatical structure to a higher degree than the language of the emails. An example dataset with suitable language could be used to demonstrate the potential accuracy of the system. Such example datasets are available from IBM's website with accompanying tutorials. However, WKS has already been shown to work well with these data sets, and experimentation with this data is not likely to lead to findings that are relevant to AVO Consulting's work.

To accurately adapt an ML model for annotation in a specific domain, knowledge of the domain language and knowledge of what parts of the text that are relevant is important. In the shipping case, the ground truth was defined based on assumptions regarding what parts of information in the emails are important to shipbrokers. The subtypes of the DATE entity were rejected in the final version of the model. This lead to higher accuracy in the annotation of this entity, but might also lead to crucial information being lost. The subtypes might be important in order to distinguish between estimated dates and confirmed data. As a best practice, the ground truth should be established in cooperation with the subject matter experts to capture all parts of the data that are relevant.

A weakness of IE systems using supervised learning is that the ground truth, defined by people, is assumed to be perfect. This is not always the case, as natural language often contains forms of ambiguity that are impossible for even people to process. A way to deal with uncertainty is to provide the confidence and the supporting evidence of each of the ML model's decisions for evaluation, e.g., by flagging instances where the ML model had particularly low confidence. This is not a feature of WKS. Instead, WKS uses the ground truth as the only measure of the degree of success, thereby basing the statistics of performance entirely on annotations made by people. An ideal IE system should produce ML models and algorithms that are as "transparent" as possible, making them easy to understand and improve by real world users.

Rule creation for NLP typically requires much more manual effort than the training of an ML model. However, rules are more transparent and therefore easier to comprehend and evaluate than the trained algorithms of an ML model. Another advantage of rules is that their output is in fixed patterns, which means there is usually no need for post-processing of the linguistic data structure found by rules. The output of an ML model, on the other hand, requires post-processing to give it meaning. For example, a date entity annotated by an ML model may have the form; "5th of June 2017", which may need to be converted to a machine-readable format like; "05.06.17" for further processing.

WKS provides rule-based tools for annotation of entity types through UIMA. One of the tools is simply called rules, and can utilize POS-tags in its conditions. This makes it possible to define complex and robust rules, especially when combined with the other tools; dictionaries and regexes. The tools can be used to pre-annotate documents, thereby accelerating the process of annotating documents for the ground truth. When a human annotator begins work on new documents that were pre-annotated, many mentions can already be annotated based on rules, dictionaries and regexes. The human annotator thus has more time to focus on assigning entity types to mentions that require deeper analysis. The rule-based tools are also easy to get started with, while ML can be used to scale the system over time. However, the final, automatically generated annotations in WKS are exclusively produced by the ML model. This is not always the best approach for all types of IE problems. Mentions with linguistic patterns that can easily be recognized by rule-based annotators may be best handled accordingly. Trying to teach an ML model these patterns may often be redundant and produce overly complex systems.

IE systems are traditionally perceived as either completely rule-based or completely ML-based. A combination of the two techniques may serve as the best solution for many IE tasks, including the tasks of the shipping case. Current systems for analyzing shipping industry emails utilize regexes to an estimated accuracy of 85%. With more available data for training, and a combination of rules and ML, the proposed system may be able to outperform the systems used today.

4.2 Conclusions and Further Work

By adding more training data and making slight adjustments in the GTE, significant improvements of the ML model were achieved. This reflects the importance of having large amounts of training data, and that the work of defining the ground truth is a crucial part of the process of making an accurate IE system. The size of the corpus used in the case was limited by the availability of relevant data. Development of sufficient training data for a practical application requires much more data, and should be carried out by multiple people.

Even with more training data, there is a chance that this application would not perform well enough to be applied to real tasks. The language of shipping emails is in a format that may prevent the utilization of some of WKS's linguistic rules processing. The tokenization, capitalization and sentence segmentation in the emails are widely different from the common rules in the English language. This makes it more difficult for WKS to recognize entity types, relation types and coreference chains, since it is unable to fully benefit from the built-in processing of linguistic structure. The challenges of the language in this case imply that better result may be achieved with rule-based methods. As discussed, a hybrid approach between ML and rules may serve as the best solution.

The rule-based tools of WKS are utilized for pre-annotation of documents in the shipping case. However, WKS also has an option for deploying rule-based models for stand-alone annotation. This feature is currently only meant for experimental use, since it is still under development. Further work with the shipping case could be to implement rule-based models for annotation of some, or all of the entity types, and compare the results to the ML approach. For the PORT and VESSEL entity types, the tools would require robust dictionaries and carefully constructed conditions to be able to annotate the correct names in the text. The DATE entity may be better suited for rule-based annotation, as demonstrated by the pre-annotation done with the rules in the case. With further development of these rules, the annotator may be able to annotate the DATE entity type entirely on its own. With a portion of entity types successfully being annotated by rule-based models, the ML model may be used as a supplement to annotate remaining entity mentions, as well as relation mentions and coreferences chains. Some of the emails in this case mention several chartering plans, and for this reason the ML model may be

required to annotate relationships and coreferences to differentiate between the different charactering plans.

Complete implementation of the IE system means taking full advantage of the automatic annotation function. To do this, the ML model needs to be made available to external applications that may utilize the information it provides. First, the model needs to be deployed in WKS. The WKS service is hosted on Bluemix, which is the cloud platform for IBM. Bluemix can show the instance name and the port name for a deployed ML model. These IDs are needed as an API call in the relevant application to gain access to the ML model. The API call can look for known entity types and relation types in a passed string of text, and return mentions and relationships recognized by the ML model. By setting up an API call like this, the ML model can be evaluated by its ability to annotate new documents.

This thesis is meant to serve as a theoretical foundation and a feasibility study of the use of ML in systems for extracting information. The area of use for systems like WKS is limited to corpora with a certain degree of linguistic structure. Rule-based approaches should always be considered in new applications of IE. With WKS, applications for IE can easily be adapted to specific domains. One of the reasons for this is that the supervised learning technique allows for annotation and training carried out by everyone, not just IT experts. ML has many advantages when it comes to processing of natural language, and it will inevitably play a key role in the future of NLP. With NLP systems supporting the Norwegian language, ML can potentially be utilized in many processes relevant to AVO Consulting.

Appendix A

Acronyms

RPA Robotic Process Automation

ML Machine Learning

NLP Natural Language Processing

MT Machine Translation

WSD Word Sense Disambiguation

POS Part-of-Speech

SVM Support-Vector Machine

NER Named Entity Recognition

CO Co-reference Solution

RE Relation Extraction

HMM Hidden Markov Model

MEMM Maximum-Entropy Markov Model

CRF Conditional Random Field

WKS Watson Knowledge Studio

regex Regular Expression

GTE Ground Truth Editor

UIMA Unstructured Information Management Architecture

SIRE Statistical Information and Relation Extraction

Appendix B

Watson Knowledge Studio Documentation

B.1 Registration

- Register with ibm.com, and create an IBM ID.
- From [IBM Marketplace](#), sign up for a free plan or purchase a subscription to use IBM Watson™ Knowledge Studio.

An instance of Watson Knowledge Studio is created for you to use. Your IBM ID is automatically given administrative access to the instance. As the administrator, you can then invite others.

B.2 Browser Requirements

The following list specifies the minimum required browser software for IBM Watson™ Knowledge Studio.

- Google Chrome 55.0.2883 and future stable fix packs (Chrome provides the best performance when using the Ground Truth Editor)
- Mozilla Firefox Extended Support Release (ESR) 45 and future fix packs

Note: If you open Watson™ Knowledge Studio in a Private Browsing window, then you must disable the tracking protection feature or allow pop-ups for Watson Knowledge Studio sites to ensure that all of the application functions will work properly.

For the best performance, use a screen resolution of at least 1024x1280.

B.3 Assembling a Team

The creation of an annotator component requires input from subject matter experts, project managers, and users who can understand and interpret statistical models. A user account must be created for each user who needs to log in to Watson™ Knowledge Studio.

About this task

Invite people to fill these roles:

- **Human annotator**

The human annotator is someone, typically a subject matter expert, who reviews domain documents to identify entities and relationships of interest to the domain. This user interacts with the application in a limited way; she uses the Ground Truth Editor to annotate a set of documents that have been assigned to her.


- **Project manager**

The project manager is someone who helps to facilitate the creation of annotator components by performing such tasks as creating project artifacts, and training, creating, and deploying models. For projects that build machine-learning annotators, they also manage the document annotation process by assigning document review tasks to human annotators, adjudicating annotation conflicts, and approving documents to add to the ground truth.

Important: If you have a free plan subscription, you cannot invite others to access your instance of Watson Knowledge Studio. Instead, you are added to the instance as an ADMIN. In the administrator role, you can perform many functions that would typically be performed by different people who are invited and assigned to different roles. You can fill the shoes of multiple people to test out how easy it is for experts from different areas of a domain to work together to build a machine-learning or rules-based model.

Procedure

To add users to a Watson Knowledge Studio system:

- 1 Log in to the My IBM Products and Services page with your IBM ID: <https://myibm.ibm.com/products-services/> 
- 2 Click **More actions** on the Watson Knowledge Studio offering tile.
- 3 Click **Manage authorizations**. To invite a user, scroll down and add the IBM ID or email address of the person that you want to invite into the **Add new user** field. If you want the user to be authorized to invite more users, then select the **Make administrator** check box. Keep in mind that most subscriptions have a user limit.

Attention: This person is given administrative access to the subscription; not to the Watson Knowledge Studio application. The user can invite or remove others from accessing the Watson Knowledge Studio instance. However, everyone that is invited, whether they are a subscription administrator or not, is automatically assigned to the HUMANANNOTATOR role within Watson Knowledge Studio. Only the admin is assigned to the ADMIN role. An ADMIN must explicitly change the user role within Watson Knowledge Studio to grant someone additional privileges.
- 4 Click **Add user**.

Repeat this process to invite more people.

The new users are sent an email, which provides information about how to complete their registration.
- 5 Now change the user roles of those you invited within the application itself. Log in to Watson Knowledge Studio with your administrator ID.
- 6 Click the settings icon to open the User Account Management page. The page lists all the user IDs that are registered as Watson Knowledge Studio users.

7

Change the user roles. By default, all users have the HUMANANNOTATOR role, this includes people that you invited to the subscription as administrators.

Important: Until you change user roles or perform some other steps, anyone who goes to the Watson Knowledge Studio instance will not be able to perform any actions or see any projects.

Click the action icon to modify the account settings. These are the available roles:

- **ADMIN**

Users in this role can:

- Manage a subscription, such as add features, storage, and more users
- Give users access to a subscription from the User Account Management page
- Create and manage all projects in their subscription
- Assign people to the project manager or human annotator roles for all projects

- **PROJECTMANAGER**

Users with this role can:

- Manage projects to which they are added
- Assign human annotators to a project

Users with this role cannot:

- Create projects
- Manage user access and roles from the User Account Management page

- **HUMANANNOTATOR**

Users with this role can annotate documents that are a.) in projects where they are assigned the human annotator role and b.) in a document set that is associated with an annotation task that is assigned to them.

Important: You must create a project, associate this user with a document set, and assign an annotation task to this user before the user will be able to see any projects listed in the Watson Knowledge Studio application. Set up the project as soon as possible after people you invited register, so that they will see your project when they first access the application. You might want to notify users after you set up the project to let them know that they can start annotating documents.

If a user has the HUMANANNOTATOR role, you can upgrade them to the ADMIN or PROJECTMANAGER roles. You cannot downgrade a user with an ADMIN or PROJECTMANAGER role to the HUMANANNOTATOR role.

- 8 | Optional: When you finish administering users in Watson Knowledge Studio , exit the session by closing the web browser. The Watson Knowledge Studio user interface does not provide an action for explicitly logging out.

What to do next

Later, if you want to remove users, log in to the My IBM Products and Services page, and from the Manage authorizations panel, click the right arrow next to the name of the user that you want to remove. Select **Remove user**, and then click **Save changes**.

Warning: If a user is assigned documents to annotate and you delete that user's account, it affects their annotations. Annotations in documents that were assigned to that user and were not promoted to ground truth are deleted.

B.4 Creating a Project

The first step in building a custom model is to create a project.

Procedure

To create a project, complete the following steps:

- 1 Log in as a Watson™ Knowledge Studio administrator, and click **Create Project**.

Note: People with the project manager role can perform almost all tasks except creating a project. An administrator must create the project initially and assign project managers to it.
- 2 Give the project a name. Choose a short name that reflects your domain content or the purpose of the annotator component.
- 3 Identify the language of the documents in your project. The documents that you add to the project, and the dictionaries that you create or import, must be in the language that you specify.
- 4 Optional: If you want to change the tokenizer that is used by the application from the default machine learning-based tokenizer, then you can expand the **Advanced Options** section, and choose **Dictionary-based tokenizer**.

The default tokenizer is more advanced than the dictionary-based tokenizer; it uses machine learning to identify the tokens in the source documents based on the statistical learning it has done in the language of the source documents. It identifies tokens with more precision because it understands the more natural and nuanced patterns of language. The dictionary-based tokenizer identifies tokens based on language rules. See [Tokenizers](#) for more details.
- 5 Optional: If you want to add project managers to the project, then expand the **Advanced Options** section, and select the names of people you want to add as project managers from the list. The administrator can add or remove project managers later by editing the project.

Only the names of people that you assigned to the project manager role from the User Account Management page for the instance are displayed. See [Assembling the team](#) for more information about adding users.

Note: If you have a free plan subscription, skip this step. You cannot add other users, so you cannot assign anyone to the project manager role. You do not need a separate project manager. As an administrator, you can perform all of the tasks that a project manager would typically perform.
- 6 Click **Create**.

Tokenizers

A tokenizer groups characters into tokens, and tokens into sentences. A token is loosely equivalent to a word.

The actions that a tokenizer must take to identify a document's tokens differ depending on the language of the document. In English, tokens are often equated to words as delimited by white spaces in a sentence. However, they do not always match one-to-one with words; other textual elements are considered tokens in some situations. For example, punctuation at the end of a sentence is considered a token, and contractions are often expanded into two tokens. In languages that do not use white spaces, such as Chinese, more complicated statistical algorithms are used to identify the tokens.

The tokenization process is important because it determines the groups of characters that users can highlight for annotation in the Ground Truth Editor. Annotations of entity and relation mentions are generally aligned with token boundaries, and must be labeled within a sentence; they cannot span sentence boundaries.

Supported types

Watson Knowledge Studio supports the following tokenizers:

- **Machine learning-based tokenizer (default)**

This is a more advanced tokenizer that identifies the tokens in the source documents based on the statistical learning it has done in the language of the source documents. This tokenizer finds tokens that capture the more natural and nuanced patterns of language. You cannot customize this tokenizer.

- **Dictionary-based tokenizer**

This tokenizer is based on linguistic dictionaries. It finds tokens that follow the rules of the source document language. Only advanced users can customize this tokenizer.

You must choose the tokenizer that you want to use when you create the project. You cannot switch to a different tokenizer later. For best results, use the default tokenizer. Only advanced users who want to modify the tokenizer behavior through a deterministic dictionary mechanism can choose the dictionary-based tokenizer. They can then customize it by adding new entries to the dictionary. However, customization must be done carefully because when you add new words to the dictionary, the changes can impact the machine-learning annotator in unintended ways.

Summary of inputs, outputs, and limitations

Different stages of annotator component development require different inputs and produce different outputs.

For each stage of the annotator component development process, this table summarizes the typical activities that you perform, the supported input file formats, the outputs that can be produced, and any sizing limits or other requirements.

All annotator component types

| Task | Typical usage | Supported input formats | Supported output formats | Limits and requirements |
|------------------------|--|--|---|--|
| Type system management | <p>Create a type system or import and modify an existing type system.</p> <p>Define entity types and relation types for your domain.</p> <p>You cannot see a visualization of the type system.</p> | <ul style="list-style-type: none"> JSON file that you exported from a Watson Knowledge Studio project ZIP file that you exported from the Human Annotation Tool (HAT) | <ul style="list-style-type: none"> JSON | <p>To avoid visual overload for human annotation, define no more than 50 entity types and 50 relation types.</p> <p>File size limitation for importing a type system: 20 MB</p> |
| Dictionary management | <p>Import a CSV dictionary file in read-only mode or a ZIP of dictionaries that you exported from another project.</p> <p>Create a new dictionary, and then import a CSV file of term entries or add term entries to it.</p> | <p>Dictionary file:</p> <ul style="list-style-type: none"> CSV file in UTF-8 format ZIP of dictionaries exported from another project <p>Term entries file:</p> <ul style="list-style-type: none"> CSV file in UTF-8 format | <ul style="list-style-type: none"> CSV file in UTF-8 format ZIP of dictionaries to use in another project | <p>File size limitations:</p> <ul style="list-style-type: none"> 1 MB per CSV term entries file 16 MB per CSV read-only dictionary file 15,000 entries per dictionary, except a read-only dictionary 64 dictionaries per project |

Machine-learning annotator component

| Task | Typical usage | Supported input formats | Supported output formats | Limits and requirements |
|-------------------------|---|---|---|--|
| Document management | <p>Import a small, representative subset of documents</p> <p>Import documents that contain annotations previously added by a human annotator, a machine-learning annotator, or a UIMA analysis engine</p> <p>You cannot ingest the entire corpus from IBM Watson Explorer for calculating high value documents for annotation.</p> | <ul style="list-style-type: none"> • CSV file in UTF-8 format • DOCXML file in UTF-8 format • Text in UTF-8 format • ZIP file that contains documents exported from another corpus • ZIP file that contains documents in UIMA CAS XMI format | <ul style="list-style-type: none"> • ZIP archive file of documents | <ul style="list-style-type: none"> • 40,000 characters per document • 10,000 documents per project • 1,000 document sets (including annotation sets) per project |
| Training and refinement | <p>Train a supervised machine-learning annotator to extract domain-specific information from unstructured text.</p> <p>Evaluate and improve a supervised machine-learning annotator.</p> <p>You cannot create a semi-supervised or unsupervised machine-learning annotator.</p> <p>You cannot do extensive feature engineering.</p> | Not applicable | Machine-learning model | <ul style="list-style-type: none"> • 1 machine-learning annotator component per project • 10 annotator component versions per project • Maximum number of projects is determined by your subscription plan. • The maximum number of training actions you can perform per month is determined by your |

| | | | | |
|-------------|--|----------------|--|--|
| Publication | Publish a machine-learning annotator to use for performing text extraction in other Watson applications. | Not applicable | <ul style="list-style-type: none"> • model ID (for use in AlchemyLanguage or Watson Discovery services) • ZIP file (for use in IBM Watson Explorer) | <p>To deploy into AlchemyLanguage, you must have a valid AlchemyLanguage Advanced plan key ID.</p> <p>To deploy to Watson Discovery service, you must know the service IBM® Bluemix® space and instance names.</p> |
|-------------|--|----------------|--|--|

Rule-based annotator component

| Task | Typical usage | Supported input formats | Supported output formats | Limits and requirements |
|-------------|--|--|--------------------------|--|
| Rule editor | Create or import documents to the Rules editor from which to define classes, regular expressions, and rules. | <ul style="list-style-type: none"> • Plain text (added in editor) • CSV file in UTF-8 format • Copied from the All document set | None | <ul style="list-style-type: none"> • 1 rule-based annotator component per project • 5,000 characters per document • 100 documents per project • Maximum size of document title is 256 characters • 200 rules per project • 400 classes per project |

| | | | | |
|-------------|--|----------------|--|---|
| | | | | <ul style="list-style-type: none"> • 100 regular expression group per project • 100 regular expression entries per regular expression group • 1,000 characters per regular expression entry • 5 rule annotator model versions per project |
| Publication | Publish a rule-based model to use for performing pattern recognition in other Watson applications. | Not applicable | <ul style="list-style-type: none"> • model ID (for use in AlchemyLanguage or Watson Discovery services) | <p>To deploy into AlchemyLanguage, you must have a valid AlchemyLanguage Advanced plan key ID.</p> <p>To deploy to Watson Discovery service, you must know the service IBM Bluemix space and instance names.</p> |

Bibliography

- Abend, O., Reichart, R., and Rappoport, A. (2010). Improved unsupervised pos induction through prototype discovery. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1298–1307. Association for Computational Linguistics.
- Allen, J. (2003a). Speech recognition and synthesis.
- Allen, J. F. (2003b). Natural language processing.
- Appelt, D. E., Hobbs, J. R., Bear, J., Israel, D., and Tyson, M. (1993). Fastus: A finite-state processor for information extraction from real-world text. In *IJCAI*, volume 93, pages 1172–1178.
- Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc."
- Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.
- Chieu, H. L., Ng, H. T., and Lee, Y. K. (2003). Closing the gap: Learning-based information extraction rivaling knowledge-engineering methods. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 216–223. Association for Computational Linguistics.
- DeJong, G. (1982). An overview of the frump system. *Strategies for natural language processing*, 113:149–176.
- Freitag, D. (1997). Using grammatical inference to improve precision in information extraction. In *In ICML-97 Workshop on Automation Induction, Grammatical Inference, and Language Acquisition*. Citeseer.

- Freitag, D. (2000). Machine learning for information extraction in informal domains. *Machine learning*, 39(2):169–202.
- Fukuda, K.-i., Tsunoda, T., Tamura, A., Takagi, T., et al. (1998). Toward information extraction: identifying protein names from biological papers. In *Pac symp biocomput*, volume 707, pages 707–718.
- Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning 2nd edition*. New York: Springer.
- Hawkins, D. M. (2004). The problem of overfitting. *Journal of Chemical Information and Computer Sciences*, 44(1):1–12. PMID: 14741005.
- Hoemsnes, A. (2017). Gjensidige vil kaste ut konsulenter og hente inn «it-hoder». *Dagens Næringsliv*. Retrieved from <https://www.dn.no>.
- Huang, F., Ahuja, A., Downey, D., Yang, Y., Guo, Y., and Yates, A. (2014). Learning representations for weakly supervised natural language processing tasks. *Computational Linguistics*, 40(1):85–120.
- IBM (2016). Watson knowledge studio documentation. <https://www.ibm.com/watson/developercloud/doc/wks/index.html>. Accessed: 2017-05-15.
- Joachims, T. (2002). *Learning to classify text using support vector machines: Methods, theory and algorithms*. Kluwer Academic Publishers.
- Jurafsky, D. (2000). *Speech & language processing*. Pearson Education India.
- Klein, D. (2005). *The unsupervised learning of natural language structure*. PhD thesis, Stanford University.
- Lacity, M. C. and Willcocks, L. P. (2016). A new approach to automating services. *MIT Sloan Management Review*, 58(1):41.
- Lafferty, J., McCallum, A., Pereira, F., et al. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289.

- Leek, T. R. (1997). *Information extraction using hidden Markov models*. PhD thesis, University of California, San Diego.
- Liddy, E. D. (2001). *Natural language processing*.
- Manning, C. D., Raghavan, P., Schütze, H., et al. (2008). *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.
- Manning, C. D., Schütze, H., et al. (1999). *Foundations of statistical natural language processing*, volume 999. MIT Press.
- Marinetraffic (2007). Port and vessel database. <https://www.marinetraffic.com/>. Accessed: 2017-02-06.
- McCallum, A., Freitag, D., and Pereira, F. C. (2000). Maximum entropy markov models for information extraction and segmentation. In *Icml*, volume 17, pages 591–598.
- Michael, L. (2016). The horsepower of hadoop: Fast and flexible insight with results. Technical report, Aberdeen Group.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.
- Miller, S., Fox, H., Ramshaw, L., and Weischedel, R. (2000). A novel use of statistical parsing to extract information from text. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 226–233. Association for Computational Linguistics.
- Mims, C. (2016). Automation can actually create more jobs. *The Wall Street Journal*. Retrieved from <https://www.wsj.com>.
- Piskorski, J. and Yangarber, R. (2013). Information extraction: past, present and future. In *Multi-source, multilingual information extraction and summarization*, pages 23–49. Springer.
- Rice, M. (2017). Watson knowledge studio workshop. personal communication. March 29, 2017.

- Shewan, D. (2017). Robots will destroy our jobs – and we're not ready for it. *The Guardian*. Retrieved from <https://www.theguardian.com>.
- Shilakes, C. C. and Tylman, J. (1998). Enterprise information portals. *Merrill Lynch, November*, 16.
- Sinha, T. and Boyd, A. (2016). Ibm watson knowledge studio – teach watson about your domain. <https://www.ibm.com/blogs/watson/2016/06/alchemy-knowledge-studio/>. Accessed: 2017-04-21.
- Somers, H. L. (1998). New paradigms in mt: the state of play now that the dust has settled. In *Machine Translation Workshop, ESSLLI*, volume 98, page 12.
- Staines, R. (2016). Ibm watson joins norway cancer cluster. *Pharmaphorum*. Retrieved from <https://pharmaphorum.com>.
- Sutton, C. and McCallum, A. (2006). An introduction to conditional random fields for relational learning. *Introduction to statistical relational learning*, pages 93–128.
- Zhu, H., Raghavan, S., Vaithyanathan, S., and Löser, A. (2007). Navigating the intranet with high precision. In *Proceedings of the 16th international conference on World Wide Web*, pages 491–500. ACM.