Virtual Assistant Design for Water Systems Operation

by

Yousra Mohamed

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2019

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

Supervisor:	Derek Rayside Associate Professor, Dept. of Electrical & Computer Engineering, University of Waterloo
Internal Member:	Mark Crowley Assistant Professor, Dept. of Electrical & Computer Engineering, University of Waterloo
Internal Member:	Victoria Sakhnini Associate Director of Software Engineering, School of Computer Science, University of Waterloo

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Water management systems such as wastewater treatment plants and water distributions systems are big systems which include a multitude of variables and performance indicators that drive the decision making process for controlling the plant. To help water operators make the right decisions, we provide them with a platform to get quick answers about the different components of the system that they are controlling in natural language.

In our research, we explore the architecture for building a virtual assistant in the domain of water systems. Our design focused on developing better semantic inference across the different stages of the process. We developed a named entity recognizer that is able to infer the semantics in the water field by leveraging state-of-the art methods for word embeddings. Our model achieved significant improvements over the baseline Term Frequency - Inverse Document Frequency (TF-IDF) cosine similarity model.

Additionally, we explore the design of intent classifiers, which involves more challenges than a traditional classifier due to the small ratio of text length compared to the number of classes. In our design, we incorporate the results of entity recognition, produced from previous layers of the Chatbot pipeline to boost the intent classification performance. Our baseline bidirectional Long Short Term Memory Network (LSTM) model showed significant improvements, amounting to 7-10% accuracy boost on augmented input data and we contrasted its performance with a modified bidirectional LSTM architecture which embeds information about recognized entities.

In each stage of our architecture, we explored state-of-the-art solutions and how we can customize them to our problem domain in order to build a production level application. We additionally leveraged Chatbot frameworks architecture to provide a context aware virtual assistance experience which is able to infer implicit references from the conversation flow.

Acknowledgements

I would like to express my sincere gratitude to the One who granted me an ever-supportive husband, an endlessly believing mom, and a little boy who continues to bring sunshine into my life everyday. Their support, patience, and sacrifice kept me going.

I would also like to acknowledge Prof. Rayside and the team at EMAGIN Clean Technologies for providing me with the opportunities and tools required to complete this work.

Dedication

To a sustainable future.

Table of Contents

List of Tables

List of Figures

Abbreviations

AI Artificial Intelligence 8, 9

ANN Artificial Neural Network 29

BICERE Boosted Intent Classification using Embedded Recognized Entities ix, x, 30–33

biLSTM bidirectional Long Short Term Memory Network ix, 2, 15, 23, 25, 30–33

BOW Bag-of-Words 28, 29

CBOW Continuous Bag Of Words 22

CI Contextual Inquiry 4

CNN Convolutional Neural Network 28, 29

CRF Conditional Random Fields 15

GVSM Generalized Vector Space Model 20, 25, 27, 33

HMM Hidden Markov Models 15

IDF Inverse Document Frequency 17

LSTM Long Short Term Memory Network x, 23, 24, 29, 30

MOE Misspellings Oblivious Word Embeddings 23, 34

NB Naive Bayes 15

NER Named Entity Recognition viii, ix, 10, 13–17, 22, 25, 27, 29, 34

NLP Natural Language Processing 15, 22, 24

NLU Natural Language Understanding 10

NP Noun Phrases 16

NTF Normalized Term Frequency 17, 20, 25, 27, 33

POS Part of Speech 14, 22

RNN Recurrent Neural Network 22, 24, 28, 29

SCADA Supervisory Control and Data Acquisition 1

SVM Support Vector Machine 15

TDNN Time Delay Neural Network 22

TF Term Frequency 17, 20

TF-IDF Term Frequency - Inverse Document Frequency 16, 28, 29, 33

TPU Tensor Processing Unit 27

VA Virtual Assistant 4

VSM Vector Space Model 20, 25

WS-LDA Weakly Supervised Latent Dirichlet Allocation 16

Chapter 1

Introduction

Water management systems are often big systems composed of multiple stages of treatment or made of complex networks wherein a multitude of variables interact to produce the overall behaviour of the system.

An example of water management systems is a water distribution system, composed of water sources, reservoirs, pumps, and piping systems. Piping systems connect reservoirs to end consumers in a specific area to meet their daily needs of water consumption. End consumers needs vary from day to day and hour by hour due to a number of factors including weather, work schedules and whether there are public events being held such as local games and festivals. Water operators have to make sure enough water is pumped into reservoirs to make sure water demand in different parts of the city is met.

Water distribution plant operators work to optimize the cost and safety of this process by choosing the right time and amount to pump water into reservoirs while taking into consideration the factors affecting water demand. In addition to that, a surplus of performance indicators are continuously being monitored by the system, and presented in a Supervisory Control and Data Acquisition (SCADA) system.

In a large system where there is an immense amount of information, it becomes a multi-click and page scan process to get the information that you need. However, process operators may want to get quick information that would help them in making better decisions for plant operation.

For that purpose we propose a water systems virtual assistant design where users can ask questions about system indicators in the most convenient way- in their local language. Through this application, users are empowered to ask higher complexity questions where

they ask about an indicator in the past or the future (e.g. Pump 1 pressure at 5 pm yesterday, or Average flow in the past two days). Under that framework, users are enabled to ask statistical questions as well as get information about forecast values that would allow them to make better decisions.

Chatbot framework architecture comprises of three different main stages; Intent classification, named entity recognition and fulfillment. In our work we dissected each of those sub-tasks in order to customize it to water systems operations queries to get better semantic inference of text in the domain.

We start our design by a field Assessment (Section 2) where we explore the work environment of the users to draw conclusions about needed features as well as restrictions that need to be taken into consideration.

In (Section 3) we provide a history of virtual assistant development and discuss the overall architecture of the system developed including its sub-components and how they are connected. In (Section 4), we describe the different approaches for building a Named Entity Recognizer including rule-based and learning-based approaches, with its variants of supervised and unsupervised learning approaches. We discuss the suitability of applying each of the aforementioned approaches and propose a model based on word embeddings which provides semantic inference capabilities to the system. Our named entity recognition system also scales up with the size and the domain of the system, allowing our users to ask about more high level concepts in addition to specific entities without having labelled data available for training.

In (Section 5), we describe how we developed an intent classifier and boosted its performance by incorporating results from the named entity recognition stage. In the section we propose and report accuracy boosts for two proposed models which improve on a baseline vanilla biLSTM.

On top of that, we conducted research in a wastewater plant, studied the environment and identified our users' needs in order to provide them with a user experience that is tailored to their needs.

Chapter 2

Field Assessment

This section investigates the design of a virtual assistant in the water process control domain, exploring the questions of the user in terms of structure and topics. We hypothesize that among the basic question structures that our system needs to capture are questions regarding performance indicators of the system, inquiries about external indicators that are relevant to process control, and asking for likelihoods on certain events happening. The study will be limited to the back-end design of the virtual assistant with plans for the front-end design to be investigated separately in the near future.

The main research queries of the back-end design of the virtual assistant are outlined as follows:

1. Exploring and establishing the benefit of a virtual assistant for water systems control.
2. Exploring the set of features that can be offered.

2.1 Contextual Design

(Beyer and Holtzblatt, 1999) [6] define Contextual Design as a state-of-the-art approach to designing products directly from a designer‘s understanding of how the customer works. They add that contextual inquiry is an effective method to avoid arguments between members in the design team where each person hypothesizes what the customer wants based on their personal preferences. Contextual design unifies design, marketing, delivery and support while catering to customers needs. Contextual Design is a six-step process: contextual inquiry, work modeling, consolidation, work redesign, actual design, and specification

[6]. We adopt the aforementioned approach focusing in this section on the first two steps; Contextual inquiry, and work models.

2.1.1 Contextual Inquiry

Contextual Inquiry (CI) is a design research method, centered at finding out the customer‘s wants and needs. More specifically, it is carried out to identify customers, know their habits, and identify how they work on a daily basis through a relaxed personal interaction with the user. CI is considered as a building block in Contextual Design, which aims to avoid the common mistake in software design where the user may get what they needed but not what they wanted. CI is built upon four basic principles: Context, Partnership, Interpretation, and Focus.

The "Context" element of contextual inquiry requires that the interviews with users are to be done in their workspace, where the researcher would be able to note different relevant aspects that can be included in the design. This may include the physical structure of the place, or perhaps some artifacts that are being generated or used by the users while they work. "Partnership" demands that the researcher collaborates with the user to be able to clearly define the user needs. The interview alternates between observing the user at work, and discussing the reasons behind different actions that they make. "Interpretation" is concerned with the conversation between the researcher and the user where the researcher confirms their understanding and insights with the user. The user either confirms the notion and perhaps expands on it, or they may correct them. "Focus" states that the researcher should be mindful of the scope of the work and should guide the conversation accordingly if it starts to drift away [30]. Contextual inquiry is based on five work models, from which we apply: physical model, artifacts model, and cultural model [6], following in the next section.

2.1.2 Physical Model

Physical model is concerned with the physical aspects of the workspace which may be relevant to our product design. This may include how the desks in the single office of the users may be aligned, noise, or perhaps the screens setup for each user.

In our visit to the wastewater plant the most prominent feature in the control room was noise. With noise outside the control room so loud that human speakers need to shout to let close-by people hear them, inside the control room the noise was barely muffled. Noise was noted and recorded to be used in testing of the Virtual Assistant (VA), and for

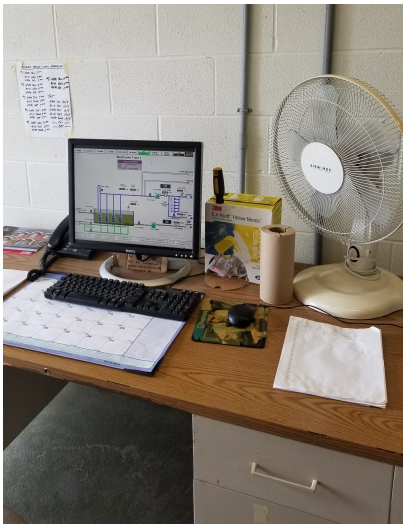


Figure 2.1: Simple workstation setup with a single screen for water operator

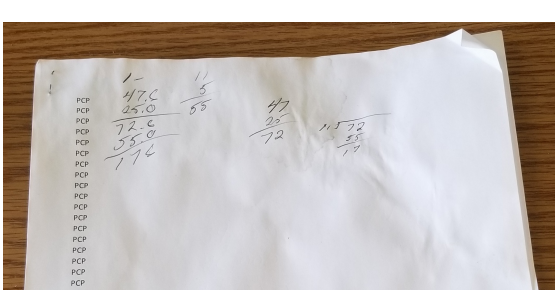
communication with customer facilities about the work-space environment requirements for our system to work properly. In the same visit it was noted that the operator uses a single screen desktop, despite having to control multiple sets of variables, and having to monitor a large set of performance indicators (Figure 2.1).

Our previous notion was that operators always work on multiple screens (typically six) to be able to monitor all the parameters effectively. This finding confirms the need for the virtual assistant and may impose some restrictions on the user interface design, which should consider space limitations while integrating the VA questions, and answers interface.

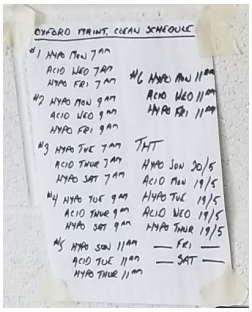
2.1.3 Artifacts Model

Artifacts model relates to the physical objects that are being used to support the user‘s work. This may typically include calendars, planners, and pen and paper, among others. The wastewater treatment utility control room contained two noteworthy artifacts; The first thing we noted is that the operator seemed to do some summations by hand using pen and paper (Figure 2.2a). This observation gave us insight that perhaps the user may be asking the VA about summaries of a specific performance indicator across different machines in the facility.

Our second observation was a schedule that was hung up on the wall (Figure 2.2b). When we asked the operator what the schedule was for he said that this is the default



(a)



(b)

Figure 2.2: On the left, figure of summation calculations done by pen and paper and on the right default control schedule hung on the wall

control parameters schedule in case the system crashes and they have to set all parameters from scratch. This observation helps us in marketing our whole smart water management solution as opposed to the old system currently in use that does not even keep memory of the default schedule for the user.

2.1.4 Cultural Model

The cultural model in contextual design entails that the researcher notes the social aspects of the work environment including any pressures that may exist, or competitiveness, and how comfortable the working environment is, among others. In our visit to the wastewater treatment facility the operator started by telling us a brief about himself, and how being only a part-timer in the facility fitted him well in his lifestyle. First thing we noticed is that he was not attached to the workplace, and furthermore we sensed a general unease in his talk about the facility. This aspect was most prominent when we visited the chemical oxidation room where two containers of very invasive chemicals; Sodium Hypochlorite (Bleach), and Citric Acid were aligned side by side in a room with no drains (Figure 2.3). The operator then made the following statement: "having the two next to each other like that makes me think of the aspect of having a possible bomb on the facility. Nothing is a 100% right?". At a different occasion when we were visiting the secondary clarifier, a member of the group that was touring the facility made a remark about the foul smell at this spot, to which the operator replied: "Well I grew numb to the smell, but I wouldn't like to live here!". We also noted some specialized mental health postings in the lunch room, which confirmed our observations about the stress level that may be implied by working in the facility.

Figure 2.3: Invasive chemicals set next to each other in chemical oxidation room

Those observations indicate the need for integrating a a virtual assistant for companionship feature wherein we incorporate small talk features, in addition to other features such as a overviews of the system given at different points in time without a user trigger.

2.2 Summary

Based on our visit to this facility, we conclude that we need to implement features to answer queries involving system indicators, and the cultural model indicates the need for personal interaction features, such as small talk and scheduled summaries. We additionally decided to provide the users with a feature to ask queries in text since some facilities may be too noisy for a speech-to-text module to interpret words properly. Moreover, in the case of text queries, spelling mistakes may occur and thus we need to take account of that in our textual analysis design.

Chapter 3

Solution Architecture

Computer scientists long aspired to build virtual assistants which best emulated human behaviour, starting from recognition of spoken words and translating it to text, followed by semantic analysis of the input and finally responding to users in the most intuitive manner. In this section, we contrast some of the most early developments in the field of virtual assistant development against the most recent developments. We additionally describe the architectures of virtual assistant frameworks which inspired the architecture which encloses different components of our solution.

3.1 History

In the early years of computing, Computer Machinery and Intelligence (Turing, 1950)[41] laid down some of the main theoretical foundations of modern Artificial Intelligence (AI). In the paper, Turing posed the question of "can machines think?" and he introduced a test whereby a machine demonstrates intelligence that is indistinguishable from a human. The test is formulated such that if a human evaluator communicates through text with two participants with the knowledge that one participant is a human and the other is not, if they were not able to reliably distinguish the human from the machine, then the machine is said to have passed the test. One of the first prominent attempts to pass the test was ELIZA (Weizenbaum, 1966)[67]. Although ELIZA failed the test, it introduced some sub-components that are a part of modern chatbot architecture, such as the recognition of keywords, specific phrases, and pre-programmed responses. Numerous other robots with comparable performance were introduced in subsequent years (Colby et. al, 1972; Wallace, 1995; Hoffer et. al, 2001)[12][65][26].

One of the rudimentary introductions in the field as well was the IBM Shoebox algorithm [23], introduced in 1961 which could recognize the spoken ten digits as well as some arithmetic operations to print out the result of those operations. HARPY speech recognition system [40] was introduced by (Lowerre, 1976). The system could recognize 1000 words- the average number of words learned by a three year old- by exploring the best acoustic paths in a state transition system.

As virtual assistants always resembled one of the top challenges for the computer industry and to the AI research community in general, machine learning research continued to take big strides through the following years. A later major breakthrough in the field was the IBM Watson system [33], introduced in 2011 which is a deep question answering system built over multiple layers of recognizing evidence in user questions, generating hypotheses and evaluating likeliness of correct answers. In the same year, Apple Inc released their world renowned virtual assistant, Siri which exhibited a near-human communication experience, including features for responding to small talk and answering simple questions over the web. Competitive companies followed in Siri's footsteps shortly after, as Google, Microsoft, Amazon and Samsung introduced their own virtual assistants Google Assistant (2012), Cortana (2014), Alexa (2015) and Bixby (2017) respectively in the following years.

However arguably the greatest breakthrough so far was the introduction of Google Duplex (2018)[36], where the virtual assistant made live calls with humans to book appointments, while handling complex situations where the person taking the call would talk distractedly, or have a non-native accent. In return, the virtual assistant in those calls displayed seamless human interaction as it hesitated, and kept good context of the conversation to infer ambiguous references. This kind of behaviour and its live test in a call with a human stirred up a big discussion on whether this interaction passes the Turing test.

3.2 Virtual Assistant Frameworks Architecture

Analysis of natural language in virtual assistants is built over four stages (Figure 3.1); Tokenization is the preprocessing stage as terms get extracted from the user question based on some regular expression pattern while getting rid of any unneeded symbols. Next step is the named entity recognition stage as terms referring to significant subjects and objects in the problem domain are recognized and labelled appropriately. Intent classification stage classifies the textual pattern of the query to one of the functionalities that are predefined in the system. The final stage is fulfillment, where the system responds to the user based on some behaviour that was previously defined in the system.

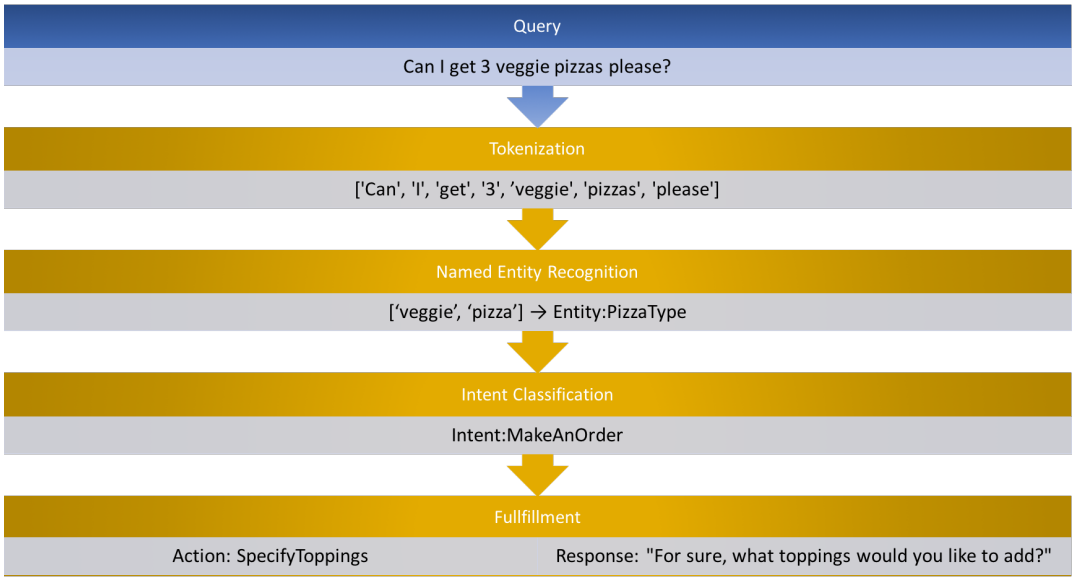


Figure 3.1: Virtual assistant stages for a pizza ordering virtual assistant

While most commercial Chatbot frameworks are not open for public inspection and provide a rather opaque view on the wiring between their different components, the stages of the architecture are still visible to us. We chose an open-source framework (Rasa) to base our design off it and improve upon its design. Rasa was compared in (Canh, 2018) against some commercial counterparts such as Google Dialogflow, Microsoft LUIS and IBM Watson Assistant, on 3 different virtual assistance datasets, and reported a competitive f1-score on those tasks. It was reported by (Davydova, 2017) that the collection of the previously mentioned frameworks is generally composed of two main components; Natural Language Understanding (NLU) and conversation flow control based on context. NLU is further broken down in the Rasa framework into two main submodules; NER and Intent Classification, which run independently after the preprocessing step (Figure 3.2).

In section 5, we explain how we modified that hierarchy by passing on information from the entity recognition stage to the intent classification stage to improve the classification performance for vague sentence structures.

The architecture of our design (Figure 3.3) demonstrates the connections between the different stages of the virtual assistant pipeline, some of their sub-components and the input connections to each of those stages. As shown in the diagram, the Named Entity Recognition stage receives the query and matches it to entities defined in the system ontology and the database while mitigating non-exact matches using edit distance and lemmati-

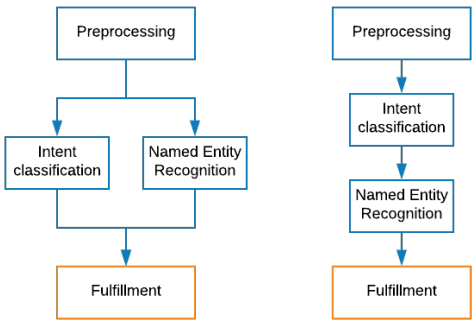
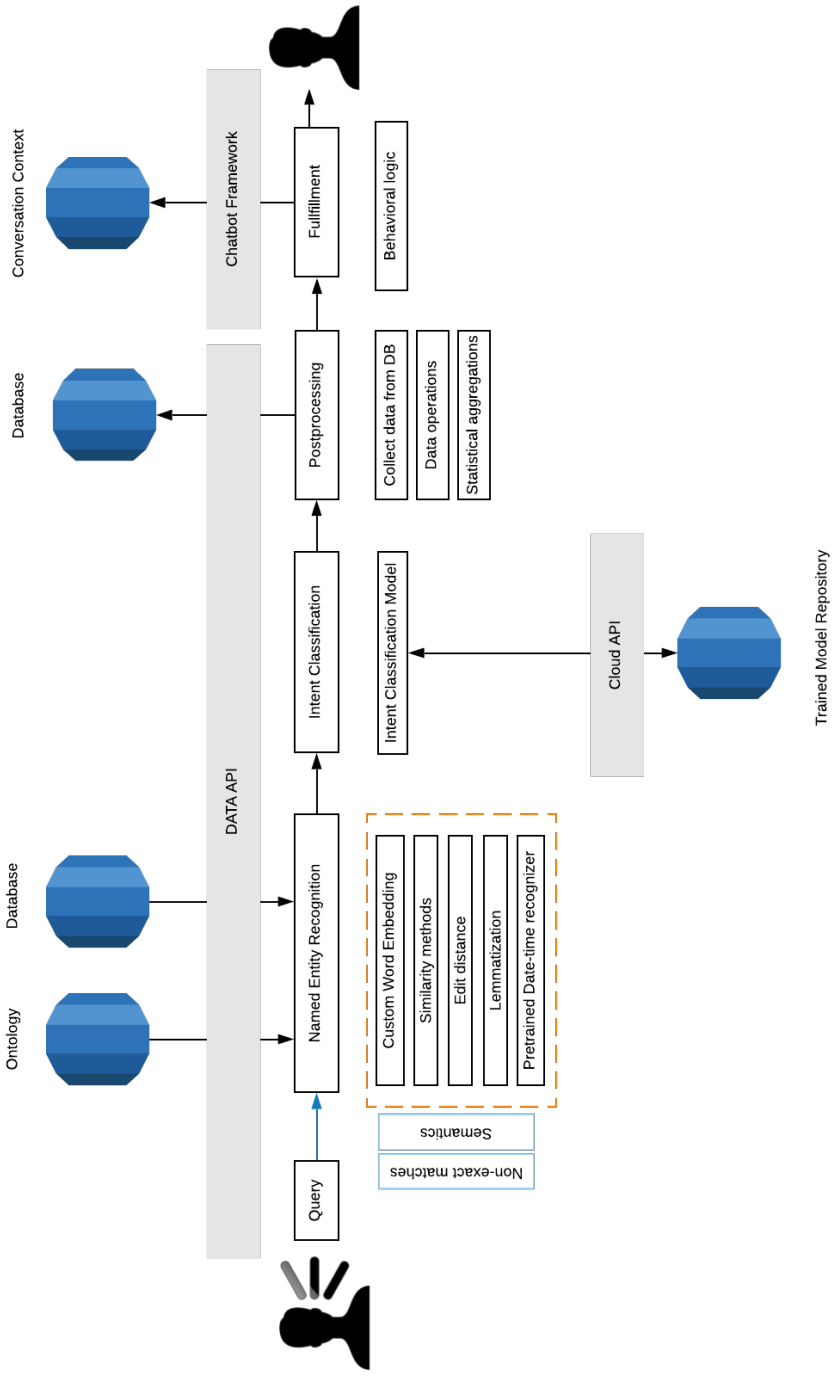


Figure 3.2: Rasa NLU architecture on left, BICERE architecture on right

zation. In the same stage, we leverage word embeddings to make semantic inferences to extract entities which are similar in meaning to defined entities but have a different lexical form. Finally we leverage pretrained datetime recognizers to extract information about time periods mentioned in the user query.

The output of the named entity recognition is then passed to the intent classification, which leverages that information to make better classifications. The post-processing stage then retrieves the information needed from the database and triggers the appropriate functions which are detected in the NER stage. Finally the fulfillment stage determines the action to take and the responds to the user with the information they need.

Figure 3.3: Overall virtual assistant architecture



Chapter 4

Named Entity Recognition

NER is the task of identifying terms of interest in unstructured text and determining the classes to which they belong. Typical entity types in general tasks that may be recognized in text include People, Countries, Organizations, ... etc. In the Chatbot framework, NER detects and classifies different entities in the user query to be able to respond in a manner that is specific to the details of the request.

Different approaches have been employed in the NER tasks, mainly split into rule-based approaches, learning-based approaches and hybrid approaches. While rule-based approaches rely more on the linguistic structure of text such as syntactic-lexical patterns to be able to detect entities and compare them to information lists (e.g. gazetteers), learning-based approaches are built over predictive models that compare the context in which the words occur and classify them accordingly.

Although learning-based approaches are powerful in the sense that they are able to address open-domain problems by training the model on a training set for a specific language then using the same model for different tasks, they are constrained in the classes of entities that they support. In order to extend a pretrained NER model to capture an extra set of classes, we would need to further train the model on new data capturing the extra classes, but this data is not always available. This restriction applies to the different types learning-based approaches adopted in NER, including supervised learning, unsupervised learning and semi-supervised learning.

Rule-based approaches on the other hand are restricted to a single domain, as information lists are crafted by domain experts in order to encompass the different entities and classes that may be significant in the domain. Syntactic-lexical patterns are also language specific which makes this approach constrained to a specific language. In the absence of

training data for NER task in a specific domain, rule-based NER has long posed as the only viable approach to tackle the problem, but recent advances in unsupervised learning for textual inference offered significant improvements to that baseline.

4.1 Rule-based NER

Rule-based approaches heavily depend on the language and domain that they are being applied to. Although rule-based approaches are open for scientists to formulate based on the language and the domain, historically they have used a finite set of approaches; Such as lexical rules, contextual rules and morphological rules of terms (Goyal et al, 2018)[24]. Those rules are often complemented by Gazetteers, which serve as a dictionary relating to a certain concept to capture patterns of interest.

For example, (Rahem and Omar, 2015)[51] applied a set of heuristics to detect names of drugs, quantities and prices, nationalities and drug hiding methods. The general method for their approach is a combination of using gazetteers for keywords related to the entity being extracted and Part of Speech (POS) tag patterns that match the entity of interest. They additionally use an algorithmic form of regular expression comparison to capture the more simple patterns such as quantities and prices. For the 5 entities being extracted, median F-measure was evaluated at 86%, median Recall at 89% and median Precision at 89%. Quimbaya et al. (2016) [50] pre-processed input text by applying stemming and lemmatization before doing a comparison to knowledge base to capture different forms of a term that revolve around the same concept. They additionally applied an edit distance metric to mitigate false negatives occurring due to spelling mistakes.

Low-resource languages have an additional interest in rules-based approach for named entity recognition. For example, (Riaz, 2010)[52] presented a rule-based approach to entity recognition in Urdu which heavily relied on lexical aspects of terms to beat a statistical based model on the same task. In their work, they discussed some of the challenges that are specific to the language and which make traditional methods incapable of encompassing the language's structure including absence of capitalization grammar rules, varying word order in a sentence and the language's agglutinative nature where certain suffixes may add complexity to the meaning of a word. They additionally discuss nested entities which are a special case of multi-word entity recognition, in addition to word reference ambiguity.

4.2 Learning Based NER

Named Entity Recognition became an intrinsic part of many Natural Language Processing (NLP) tasks in the last few years, including question answering systems [46][59], information retrieval[31], relation extraction[27], machine translation[2][10] and text summarization[4][47], among others. Due to the importance of this task, more data became available to enable machine learning algorithms to learn patterns capturing specific types of entities in open domains.

Supervised learning is one of the main approaches for performing NER, avoiding gazetteers and handcrafted rules that match a defined subset of entities. Instead, classification models form a feature space capturing the context around words to infer the class to which that term belongs.

While rule-based approaches are more suited for specialized domain entity recognition, learning based NER is much more powerful on open domain data as it generalizes its learning of the context around words to extend that knowledge to unseen data. Examples of entity classes that are typically captured by learning based approaches include persons, locations, date and time, organizations, ..etc. Most common classification models used for NER include Support Vector Machine (SVM) [3][21] , Conditional Random Fields (CRF) [37][11][7], Hidden Markov Models (HMM) [66], Naive Bayes (NB) and logistic regression[68].

Lample et al. (2016)[35] presented two deep neural architectures for named entity recognition tested on four different languages, achieving competitive to state of the art performance on CoNLL-2002 and CoNLL-2003 data sets. In the LSTM-CRF architecture that they presented, global word embeddings were passed to a biLSTM which produces for each term encodings of its context from both left to right and vice versa. The encodings from the previous step are then concatenated and passed to a CRF which performs the final classification. They additionally enhance word embeddings using a character-based word embedding in order to avoid language-specific morphological hand-crafted rules.

Speck and Ngomo (2014) [58] compared different ensemble techniques including AdaboostM1 and Bagging with J48 as base classifier as well as Random Forests against ten other single classifiers to determine the dominance of ensemble learning in NER task. The results, combined over four different datasets, showed Random Forests, Multi-layer Perceptrons, AdaboostM1 and J48 as the top-tier performers on the NER task. Among the models that were included in the study are the commonly used Naive Bayes, SVMs and Logistic regression, however CRFs and HMMs have not been included in the study.

Zhang and Elhadad (2013)[71], took an unsupervised approach to perform NER in

the biomedical sector. They start with seed terms that are extracted from external terminologies to represent different entity classes. To extract candidate entity expressions they use an Noun Phrases (NP) chunker followed by document frequency calculation to select the most common nouns as a filtration step. Finally they calculate cosine similarity between candidate expressions represented by a Term Frequency - Inverse Document Frequency (TF-IDF) signature of the terms in the expression as well as the context that they appeared in and the averaged TF-IDF representation of groups of terms representing different classes.

Guo et.al (2009) [25], introduced a novel topic model dubbed Weakly Supervised Latent Dirichlet Allocation (WS-LDA) to tag a single named entity in search engine queries. Their model maximizes the probabilities of the context around possible entities as well as the probabilities of different classes given a specific entity.

In the context of special domains, usually there is no training data available. We utilized unsupervised approaches to build our NER system. Starting with seed labeled entities, we calculated the similarity between those labels and chunks from users queries. Furthermore, we introduced a new approach in which we leveraged word embeddings' representative power to provide better semantic comparison between the queries and our knowledge base. We finally incorporated a hierarchical ontology in order to handle queries about vague concepts and answer them in more specific terms. Each of the aforementioned sub-components of our system are discussed in the following subsections.

4.3 NER for Water Systems Operation

Named Entity Recognition is the stage that enables Chatbots to address specific items in users queries and respond accordingly. For example a Pizza restaurant Chatbot would need to implement NER to detect food item entities like "Pizza". The system would need to recognize provided information about additional entities such as "toppings" and prompt the user for missing information regarding the entity of interest.

In the context of chatbot design for water systems operation we are faced by a number of challenges; Firstly, there is no training data for detecting entities specific to water systems. Consequently, we cannot utilize state of the art supervised models to detect the entities. Additionally, there are often different ways to refer to the same entity; For example, different morphological forms of the same term can be used interchangeably, such as "operational cost" and "cost of operations". Technical synonyms also often exist such as "mixed liquor" and "biological mass", and "flocculating agent" and "flocculant"

in wastewater systems. We design our NER solution starting from the finite set of entities that our system can provide information on and provide extension to unseen synonymous terms.

4.3.1 Seed Space Representation

Starting from a list of seed terms that were crafted by subject matter experts, we designed a representation space that user queries would be compared to in order to extract terms of interest. We used Normalized Term Frequency (NTF)[55] representation space to encode the seed space of terms, such that the whole space $\mathbf{X} \in \mathbb{R}^{d \times t}$, where d represents the number of documents which map to technical terms for our system and t is the total number of unique terms across all documents. In Term Frequency (TF) representation, $\mathbf{X}$ is a sparse matrix wherein x_{ij} at row i and column j is equal to the number that term j occurred in document i.

Normalization is a term weight adjustment step such that a document is represented by the weight of occurrence of its terms and normalized by the document length, such that elements of normalized matrix $\mathbf{Y}$ are calculated as:

$$y_{ij} = \frac{x_{ij}}{\sum_{j=1}^{t} x_{ij}} \qquad \text{for } i = 1, 2, ..., d \tag{4.1}$$

Normalization offers a fair method to compare occurrences of terms across documents that have different lengths to eliminate the dominance of term frequencies that comes with longer documents over shorter documents.

Term-frequency is usually coupled with Inverse Document Frequency (IDF) [53] , which re-balances term weights such that frequent terms, that have high frequency across the whole corpus get reduced weights. Typical words that get demoted using IDF are stop words [22] that show no distinguishing properties to different documents. We chose not to use IDF in our corpus as named entities are concise and rarely include stop words.

4.3.2 Hierarchical Entity Representation

In order to provide reference to more general entities, we represent entities in a hierarchical manner through an ontology representation, as shown in (Figure 4.1). In this manner we are able to break down a user's question to more specific terms such that if they ask about a parent entity we are able to break it down for them, providing information about its

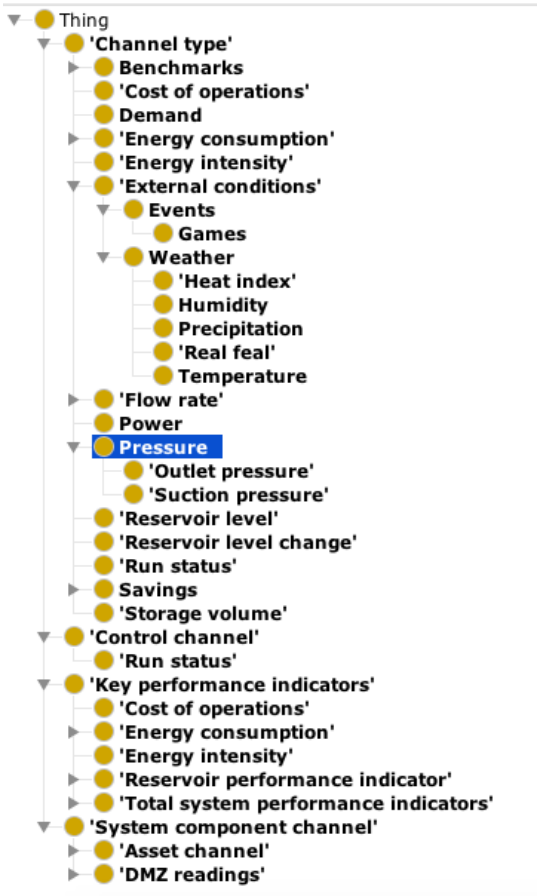


Figure 4.1: Named entities ontology hierarchy for water distribution systems

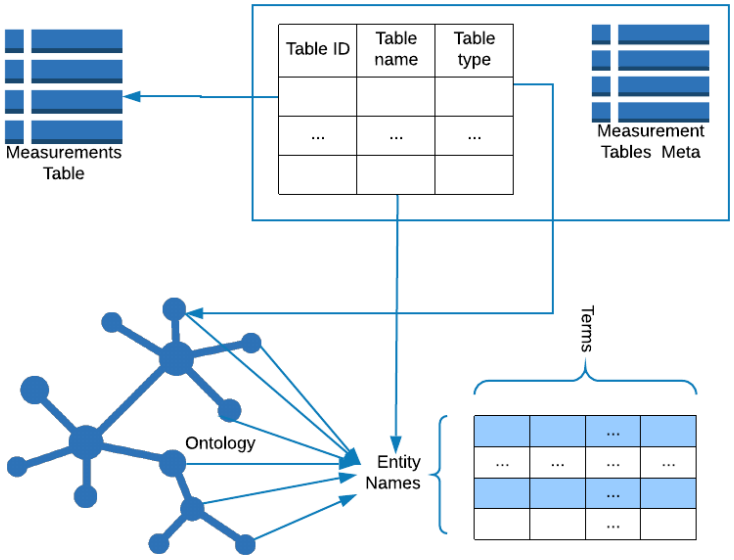


Figure 4.2: Named Entity Seed Space Representation

children. For example, if the user asks about the "Pressure", we give them options to get information about "Outlet Pressure", "Suction Pressure" or both.

Leaf entities (ontology individuals) are further mapped to tables in the database, containing real-time measurements that relate to the users' queries. We also do comparisons between users queries and table names defined in the database to match uniquely named entities stored in the database (Figure 4.2).

Ontology representation [42] was chosen to represent the hierarchy as it provides features to add information about the different classes and individuals, to provide assistance if the user is confused about a term after a prompt. Ontology representation additionally provides tools to define relationships between classes and individuals. Through the "owl:sameAs" relationship in the ontology we were able to explicitly assign synonyms when we do not have data to provide that relationship. This enables us to cater to a wider set of dialects and technical term conventions that vary from one region to the other.

Through the ontology we were also able to link measurement tables to assets through a coined relationship "hasAsset", which connects an asset to different measurements over time. When the user asks about measurements for a certain asset, we collect data about different indicators relating to the asset of question to answer the query.

4.3.3 Similarity Calculation

Different approaches for similarity calculation of text documents were contrasted, while taking cosine similarity as the benchmark.

Cosine Similarity

Cosine similarity [57] is a method to calculate the similarity, represented as the angle between two vectors such that if $\vec{x} = [x_1, x_2, \dots x_t]$ and $\vec{y} = [y_1, y_2, \dots y_t]$, then their similarity is calculated as:

$$cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\|\|\vec{y}\|} = \frac{\sum_{i=1}^{t} x_i y_i}{\sqrt{\sum_{i=1}^{t} x_i^2}\sqrt{\sum_{i=1}^{t} y_i^2}} \tag{4.2}$$

Similarly, the similarity between two documents in TF form can be calculated using cosine similarity. In order to extract named entities from users' queries, and after the users' input is processed we chunk the input into ngrams of lengths $l = 1 \dots g$.

$$g = \max_i |\vec{x_i}| \qquad \text{for } i = 1, 2, ..., d \tag{4.3}$$

Where d is the number of documents defined in the system. The ngrams are then compared against defined technical terms to match to most similar document. If the top similarity is greater than a defined cutoff threshold r then the match is kept for further comparisons, otherwise the ngram is said to have no matches.

Generalized Vector Space Model (GVSM)

In Vector Space Model (VSM), document-document similarity is calculated as $\mathbf{X}\mathbf{X}^\top$, where $\mathbf{X}$ is a NTF matrix as defined in 4.1. Wong et. al (1985) [69] criticized the false assumption of orthogonality between terms in this space model and introduced Generalized Vector Space Model (GVSM) instead which captures the relationships between different terms. GVSM introduced a term-term correlation matrix for the calculation of document-document relationship expressed as $Sim_{GVSM} = \mathbf{G}\mathbf{X}\mathbf{X}^\top$.

$$Sim_{GVSM} = \begin{bmatrix} x_{11} & x_{12} & \dots \\ \vdots & \ddots & \\ x_{d1} & & x_{dt} \end{bmatrix} \times \begin{bmatrix} g_{11} & g_{12} & \dots \\ \vdots & \ddots & \\ g_{t1} & & g_{tt} \end{bmatrix} \times \begin{bmatrix} x_{11} & x_{12} & \dots \\ \vdots & \ddots & \\ x_{t1} & & x_{td} \end{bmatrix}$$

G is a term-term correlation matrix whose elements can be calculated using cosine similarity calculated between term vectors. Term vectors may be extracted from the original space **X** of documents or other open document space using different methods including term frequency and context space represented in (Zhang and Elhadad, 2013)[71] or using unsupervised models of word embeddings.

Word Embeddings

Word Embeddings are vector representations for textual terms that sparked a revolution of natural language processing developments in recent years. Word embeddings represent words or phrases in a n-dimensional space where similar words would be close to each other. Word embeddings can also have compositional properties as was showcased in the introduction of GloVe embeddings (Pennington et al, 2014) [48], where

$$\vec{King} - \vec{Man} + \vec{Woman} \approxeq \vec{Queen} \tag{4.4}$$

Bengio et. al (2003) [5] was first to coin the term "Word Embeddings" by referencing the weights matrix of the Softmax layer in a one-hidden layer feed-forward language model neural network. i.e. The network predicts the next word w_t from the sequence of preceding words $w_{t-1}, ...w_{t-n+1}$. Taking in a d-dimensional representation for each of the preceding words in a window of size t, they are fed to a hidden layer of size h and connected to a final Softmax layer indicating the distributional probability of each of the words $w_1, ...w_N$ occurring as the next in sequence. Given a large corpus of text, the network then maximizes the combined log-likelihood of all words sequence in the dataset as

$$L(\theta) = \sum_t \log P(w_t | w_{t-1}, ...w_{t-n+1}) \tag{4.5}$$

$$\text{where } P(w_t = k | w_{t-1}, ...w_{t-n+1}) = \frac{e^{a_k}}{\sum_{l=1}^{N} e^{a_l}} \tag{4.6}$$

$$\text{and } a_k = b_k + \sum_{i=1}^{h} \mathbf{W}_{ki} \tanh c_i + \sum_{j=1}^{(n-1)d} \mathbf{V}_{ij} x_j \tag{4.7}$$

Bengio et. al (2003) highlighted that the bottleneck for training the network is the normalization calculation performed at the Softmax layer as it is proportional to the number of words in the vocabulary, which can reach the order of millions.

Collobert and Weston (2008) [13] used a similar notion to train word embeddings which are represented as a lookup table in the input of a deep neural structure. The word embeddings were tuned by consecutively integrating the lookup table to network structures trained on 5 different supervised NLP tasks, such as Semantically Related Words (SRL) task, POS tagging, Chunking, NER and Language Modelling. The rest of their network structure relies on Time Delay Neural Network (TDNN) [64], which behave like modern Recurrent Neural Network (RNN)s in a manner where at each time t the network only sees w_t while keeping context of previous words. The TDNNs are further stacked in a convolutional structure to capture local features as well as global features to keep better context of text.

Mikolov et. al (2013) [44] introduced the landmark Word2vec, which builds upon and simplifies previous models with two architectures presented; Continuous Bag Of Words (CBOW) and Skip-grams. In their paper they observe that for the purpose of training word embeddings, data is available for including context around words from both directions, yet in previous language model approaches the tuning only relied on the preceding words under the assumption that the network would be used for next-word predictions. Thus under their new framework, in the case of CBOW the network is structured to estimate $P(w_t|w_{t-n}, ...w_{t-1}, w_{t+1}, ...w_{t+n})$. In the case of Skip-grams, the equation is reversed where we estimate the surrounding context of a specific word as $P(w_{t\pm i}|w_t)$ for $i = 1, ...n$. The network additionally does not include a hidden layer, directly connecting the input to the output which significantly simplifies parameter tuning of the network, represented as the average log probability of all words $w_1, w_2, ...w_T$ in the network

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq i\leq c, i\neq 0}^{c}\log P(w_{t+i}|w_t)$$

where c is the context size around the word. In analogy with 4.5 and given that the output layer is directly connected to the input layer, the posterior probability is represented as

$$P(w_{t+i}|w_t) = \frac{\exp v_{w_t}^{\top} v'_{w_{t+i}}}{\sum_{w_j \in V} \exp v_{w_t}^{\top} v'_{w_j}}$$

where v_w and v'_w are input and output embeddings of word w, respectively and V being the size of the vocabulary.

GloVe embeddings (Pennington et al, 2014) [48] encodes information about the ratio of co-occurrence probability of different words using a least squares cost function which minimizes the difference between the dot product of vectors of two words and the logarithm

of their respective number of co-occurrences

$$J = \sum_{i,j=1}^{V} f(\mathbf{X_{ij}})(w_i^\top \tilde{w_j} + b_i + \tilde{b_j} - \log \mathbf{X_{ij}})$$

where X_{ij} denotes the number of times w_i occurs in the context of word w_j, w_i and $\tilde{w_i}$ being the word vector and context word vectors of word i respectively, and b denoting the bias for each of the vectors. Function f further down-scales the weights of too low and too high co-occurrence frequencies. GloVe word embeddings showed superior representation of words in the new space such that semantically or linguistically similar words would have a smaller euclidean distance between them and they exhibited the linear substructures as outlined earlier by 4.4.

FastText (Bojanowski et al, 2016)[8] represented words as ngrams of its substrings, allowing it to capture information about prefixes, suffixes and morphological subsequences that tie semantically related words together. For example, the word $FastText$ with ngram size $n = 4$ would be represented using ngrams ['Fast', 'astT', 'stTe', 'tTex', 'Text']. The ngrams representations are then trained using a Skip-gram model and combined as a bag of words to represent different words for which they are sub-strings. The word dissection feature of words in FastText enabled it to infer embeddings for out of vocabulary words. Misspellings Oblivious Word Embeddings (MOE) (Edizel et al, 2019) [20] utilized FastText in training on a dataset which maps misspellings to their correct variants producing a word embedding model which maps incorrect misspellings to the same space as their correct variants. MOE achieves this by adding a spelling correction loss to FastText's semantic loss which is minimized to map similar words which have similar contexts to the same space as well as their misspelled counterparts.

ELMo (Peters et al, 2018) [49] is a powerful word embedding model which produces embeddings of a word based on its morphological structure as well as its context such that it produces different word embeddings for the same word which appears in different contexts. Using sub-sequence representation of words using character ngrams, they applied a convolutional neural network, followed by a max-pool layer to be passed to a biLSTM network. Their model additionally uses a 2-layer highway network as an intermediate layer to connect the max-pool output to the LSTM input. Highway networks (Srivastava et al, 2015)[60] form information highways between different layers of a neural network by using gating mechanisms to regulate information flow. Highway networks have shown to significantly optimize the network training time allowing us to create deeper network structures which make better inferences of information. In the biLSTM layer of ELMo, a residual connection additionally connects the input of the first LSTM layer to the second layer helping the model train more successfully.

BERT's architecture (Devlin et al, 2018) [18] similarly trains the deep bidirectional transformer encoder architecture introduced by (Vaswani et al, 2017) [62] to produce contextualized word embeddings using the widely available open text data. In the training phase they pass sentences to the model, while masking 15% of the words and let the model adjust its weights to predict the missing words. The model can further be tuned on the different NLP supervised tasks by plugging in the model as a word embedding phase in a larger neural architecture.

Sentence Embeddings

In order to compare entities defined in our system composed of one or more words with sub-sequences of words from the user's queries we can represent each word as a word embedding then combine them to make the comparison. A powerful yet simple baseline is to simply average the vectors of different words belonging to the sequence of words (Conneau et al, 2018; Shen et al, 2018) [15][56]. A simple modification that showed significant improvements was showcased by (Arora et al, 2016) [1] where they got the weighted average of the individual vectors then performed common component removal where they removed their first principal component. That approach, although simple introduced significant improvements on supervised tasks amounting to 30% in some cases and beat some of the complex models built using RNN's and LSTM's. Ruckle et al. (2018) [54] introduced another simple aggregation of word embeddings in which they apply the power mean

$$(\frac{x_1^p + ... + x_n^p}{n})^{1/p}$$

to produce different power means $w^{(p_1)}, w^{(p_2)}, ...$ which are concatenated to represent the sentence. They additionally concatenate power means of word embeddings taken from different embedding sources for better representation.

Other model based sentence embeddings include skip-thoughts (Kiros et al, 2015) [32] which build upon skip grams by using the current sentence s_t to predict the surrounding sentences $s_{t\pm i}$ for $i = 1, ..c$, where c is the context. A RNN encoder-decoder model is used to make those predictions. Another model is quick-thoughts (Logeswaran and Lee, 2018) [39] in which the next sentence is chosen from a set of possible next sentences using a classifier. After the input sentence s_t and the candidate following sentences S_{cand} are encoded, they are passed to a classifier which determines the predicted next sentence s_{t+1}^p. Let f and g be parametric sentence encoding functions, and k being a scoring function of a classifier, the probability that $s_c \in S_{cand}$ is the correct following sentence of s_t can be

written as

$$p(s_c|s_t, S_{cand}) = \frac{\exp k(f(s_t), g(s_c))}{\sum_{s'_c \in S_{cand}} \exp k(f(s_t), g(s_{c'}))}$$

and the training objective then maximizes the overall probability of correct context sentences for each sentence in training data D as

$$\sum_{s \in D} \sum_{s_c \in S_{cand}} \log p(s_c|s, S_{cand})$$

InferSent (Conneau et al., 2017)[14] built a sentence encoder in the context of the sentence entailment task. The SNLI corpus is a corpus of 570k pairs of sentences labelling the relationship between a premise sentence p and a hypothesis sentence h as: entailment, contradiction or neutral relationship. At the input, InferSent encodes h and v using a biLSTM topped with a max-pool layer. The encoded sentences h' and v' are then represented using a concatenation, an element-wise product between the two, and an absolute difference between them, to be passed to multiple-layer fully connected neural net and a 3-way Softmax is finally attached to classify the relationship between the sentences.

4.4 Experiments

In our experiments, our focus was to build a NER system which is able to match to entities that we define in our ontology as well as relate similar concepts in the user's query to those entities. To provide the power of semantic inference of text while avoiding hand-crafted morphological rules, we leveraged the power of word embeddings in textual similarity calculations.

The benchmark for our experiments is the NTF representation of defined entities, calculated against token n-grams extracted from the user's query, represented in the same VSM space. Similarity is calculated as the cosine similarity between each of the documents and the token n-gram to extract the top similar entity if the similarity exceeds the threshold r_{min}. We additionally implement the GVSM model, described in 4.3.3, where g_{ij} is represented as the similarity between the embeddings of term i and term j. Another similarity measure that we used is relative euclidean distance, in which we get the euclidean distance between d_i and d_j, divided by the median of distances between all documents $d_k \in D$ in the corpus and d_i.

Sentence embeddings representing defined entities and token n-grams from user's queries are also directly compared for similarity. Sentence embeddings were represented as the av-

erage of its constituent words and using the power mean representation 4.3.3. Power mean representations for $p = [2, 4, 8]$ were concatenated together to represent the sentence.

We collected 11.2k sentences relating to water operation, scraped from a water and wastewater operators community (https://www.fluksaqua.com) to be used as training data for word embeddings. FastText word embeddings were trained on this data to yield embedding models of 150 and 450 dimensions to be used with the concatenated power mean sentence representation and the averaging sentence representation modes respectively. Learning rate was empirically tuned between 0.01 and 0.1 and epochs varied in the range $epoch = [5, 8, 11, 15]$ to yield the best word representation at $lr = 0.03$ and $epoch = 8$. Empirical testing was determined by examining the similarity between groups of similar words that were expected to be projected close to each other and have a higher separation from other groups of similar words.

A pretrained BERT embedding model which represented terms in 768 dimensions was further tuned on the collected data to give a better representation of terms in water systems. The BERT embedding representation was truncated to 150 and 450 dimensions to be used in different sentence representation modes as previously described.

Combinations of the described design variants were compared on a hand-labelled dataset of 103 utterances, which yielded a set of 1764 text chunks to be contrasted against a collection of 119 defined entities.

4.4.1 Evaluation Metrics

For each of the design combinations we report precision, recall and the f1-score represented using the following equations.

$$Precision = \frac{TP}{TP + FP} \tag{4.8}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.9}$$

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.10}$$

where TP, FP, FN are true positives, false positives and false negatives respectively. Each of the metrics reported is calculated as the weighted average over the classification results of different classes in the dataset.

				Precision	Recall	F1-score
NTF			cosine	0.748803	0.774121	0.750800
GVSM		BERT	cosine	0.939290	0.898469	0.914171
		FastText	cosine	0.939290	0.898469	0.914171
EMBED	AVG	BERT	cosine	0.939582	0.908678	0.921063
			rel-euclid	0.882477	0.838911	0.860143
		FastText	cosine	0.944107	0.908111	0.924268
			rel-euclid	0.882703	0.842314	0.862036
	POW-MEAN	BERT	cosine	0.936535	0.906410	0.918599
			rel-euclid	0.883892	0.843449	0.863197
		FastText	cosine	0.945014	0.912649	0.926536
			rel-euclid	0.881552	0.841747	0.861190

Table 4.1: NER experiments benchmarked on Precision, Recall and F1-score

4.4.2 Results

As shown in table 4.1, using word embeddings to encode term semantics shows a considerable improvement over the baseline NTF text representation, with GVSM word space representation having a significantly high performance and embedding representation having a slightly higher performance than GVSM. The results also show that cosine similarity consistently dominates relative euclidean distance similarity while varying the sentence representation between averaging and power mean representation had no effect on the performance.

It is also noteworthy that FastText is at the same level or higher performance as BERT, considering that FastText is very fast to train, while pretrained BERT had to be tuned for 10,000 steps over Tensor Processing Unit (TPU)s for several days to reach the performance reported. Although embedding results show considerable performance boost over the baseline, we believe that it can be further improved using a larger dataset that can be collected over time to capture a wider array of expressions used in the field and scale up the number of examples seen by the model.

Chapter 5

Intent Classification

Text classification is a classical problem in natural language processing where input text is cleaned, features are extracted and represented in a mathematical form- often represented as vectors or matrices- so classifiers can be applied to identify the corresponding classes. Traditional methods for feature extraction include Bag-of-Words (BOW), n-grams and TF-IDF. Linear classifiers were generally used as the topics tend to be easier to separate in higher dimensions. Since BOW, n-grams and TF-IDF miss the interaction between the different terms in a sentence, and the implication of one sentence on the next, more advanced algorithms have been implemented to address the same problem in recent years, with considerable higher accuracies.

5.1 Literature Review

Miroczuk & Protasiewicz (2018) [45] highlighted the wide breadth of text classification applications, ranging from industrial domain to medicine, reportedly spanning over 15 different domains. They also dissected the classification tasks into four categories; Binary classification, multi-class, multi-label, and hierarchical classification. Hierarchical classification yields multi-label classification in a way where each node falls into one of the classes and the multiple nodes in a path represent the multiple labels that can be assigned to an input.

Vilar (2019) [63] contrasted the performance of Convolutional Neural Network (CNN)s and RNNs, where CNNs showed better handling of feature extraction around words, while RNNs were more proficient at capturing the sequential feature patterns of text. They

further explained that although variants of RNNs (e.g. LSTMs and Gated Recurrent Units - GRUs) may be more suited to language modelling, they showed comparable results to CNN performance for short text, and better performance for long texts. RNNs being more complex, need more time for training than CNNs.

Zhang et. al (2015) [72] presented a character-level CNN for text classification. The method contrasted itself against other well-known approaches such as word-level CNNs, vanilla LSTMs with word embeddings, as well as BOW, n-grams and TF-IDF input representation with multinomial logistic regression for large-scale datasets. The proposed approach had a comparable performance across the 8 datasets presented. Zhou et. al (2015) [73] introduced the combination of CNNs and LSTMs where CNNs learn high level representations of text and LSTMs observe the sequence correlation. The proposed model outperformed separate models of CNNs and LSTMs on text classification tasks.

Joulin et. al (2016) [29] introduced an Artificial Neural Network (ANN) with 10 hidden units, bigram word representation and hierarchical labelling to be trained on large-scale datasets, achieving state-of-the-art accuracy while achieving increasing speedup with larger datasets, amounting up to $15,000\times$ compared to its counterparts. Ding et. al (2018) [19] adapted the landmark DenseNet architecture (Huang et. al, 2017) [28] to natural language processing, proposing an architecture where sequence of hidden states is considered as reading memory for each layer, and information at input layer and hidden layers are passed on to all subsequent layers. The architecture allows that memory states accumulate information with deeper layers and it is bidirectional such that layers at the same level are connected. Their proposed densely connected architecture reportedly performed better than its deep stacked counterpart.

Meng & Huang (2017) [43] introduced a double-tiered model for dialogue intent classification, formed of two LSTMs to hierarchically model user input at sentence level and conversational context level, while adding an external memory unit to improve dialogue intent classification, achieving a 2.2% accuracy boost over basic LSTM model.

5.2 NER Augmented Intent Classification

Virtual assistants are faced with a unique challenge in which the input text is short, which gives classifiers less information to incorporate in its inference. Moreover, users may phrase questions addressing different intents using the same pattern with a difference in a word or two. e.g. "Show me homepage" and "Show me sensor A" would refer to "navigation" and "sensor data" intents respectively with so little difference in the query pattern. The core difference between the two queries however are the entities and their classes.

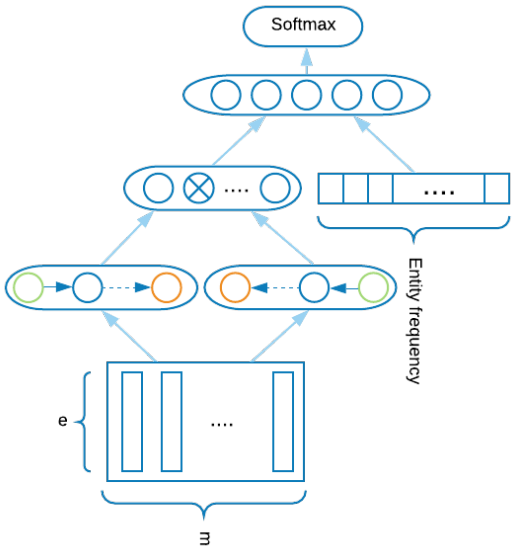


Figure 5.1: BICERE architecture; Green and orange nodes denote start and end of sequences of an LSTM

In this section we explore providing extra information for the intent classification task, which constitutes of the classes of recognized entities in the query. We contrast two models; biLSTM+ is the first model where we augment the input to a biLSTM classifier with recognized entities' classes. BICERE is the second model where we combine the results of a biLSTM and an encoding of the recognized entities' classes to improve the classification.

As shown in (Figure 5.1), BICERE takes in sentences with variable length, denoted as $S = w_1, w_2, \ldots w_r$. After data is cleaned and unique vocabulary is extracted, each sentence is represented as $S' = i_1, i_2, \ldots i_m$, where m is the maximum number of terms in a sentence and i_j points to the index of some term in the vocabulary.

Next we setup our model, such that in the first layer we embed each of the items in a sentence into a space vector, which is tuned in the training phase of the model. The embedded sentence, $S_e = e(i_1), e(i_2), \ldots e(i_m)$ is then passed to a bidirectional LSTM, which captures the structure of the sentence and keeps attention of the flow. We chose biLSTMs in this layer over unidirectional LSTMs, as they have proved their proficiency in keeping context of the sentence from start to end and vice versa. They also seem more suited to the task of text classification as the whole input sequence is known at prediction time, as opposed to language modelling tasks where only the first part of the sentence may be known. The output of the biLSTM layer is denoted as $h_s = h_1, h_2, \ldots h_{2e}$, where e is

	100 epochs			150 epochs		
	biLSTM	BICERE	biLSTM+	biLSTM	BICERE	biLSTM+
Accuracy	0.789	0.812	0.849	0.869	0.882	0.930
Validaton Accuracy	0.697	0.683	0.798	0.707	0.700	0.778
Loss	1.035	1.032	0.795	1.170	1.122	1.043
Precision	0.71	0.75	0.78	0.71	0.71	0.76
Recall	0.71	0.73	0.80	0.72	0.71	0.76
F1-score	0.71	0.70	0.78	0.71	0.70	0.76

	200 epochs		
	biLSTM	BICERE	biLSTM+
Accuracy	0.913	0.915	0.954
Validation Accuracy	0.714	0.710	0.781
Loss	1.441	1.239	1.297
Precision	0.73	0.69	0.75
Recall	0.73	0.72	0.76
F1-score	0.73	0.70	0.75

Table 5.1: Performance report for biLSTM, BICERE and biLSTM+ at 100, 150 and 200 epochs.

the embedding size of the input. h_s is then passed to a dense layer with a dropout rate of 0.5 to produce $q_S = q_1, q_2, \ldots q_e$.

Meanwhile, a second input entity frequency array f_S is passed in, such that $f_S = [r_1, r_2, \ldots r_n]$, where n is the number of entity types predefined in the system, and r_i refers to the number of times entity type i was recognized in the sentence. The final layer takes as input the concatenation $q_S|f_S$ and passes the output to a Softmax which produces a sequence $x_1, x_2, \ldots x_k$, where k is the number of intent classes and x_i denotes the probability of the input being classified into class i.

In the next section, we contrast the performance of a vanilla biLSTM, BICERE and biLSTM+ on an intent classification dataset that we collected in a demo period.

5.3 Experiments

To train our model, we collected 262 utterances as a baseline virtual assistant, built using Dialogflow framework was presented in a trade show for water technology. The entities

were defined on Dialogflow platform and users were directed to ask questions relating to 4 different intent classes. An additional intent class default.fallback was added by the framework in reference to statements that do not fall into one of the patterns of the defined classes, and 11 different entity types were found in the dataset. Results of the classification and entity recognition were manually revised to make sure correct annotations are assigned to training data.

Throughout our experiments we used embedding size $e = 128$, and the longest sentence size $m = 32$.

For each of the models, we reported the accuracy, validation accuracy and loss at 100, 150 and 200 epochs. We additionally report precision, recall and f1-score 4.8 which were calculated using a 7-fold validation scheme. Moreover, the values given for each of the results at n epochs maps to the average of the $n - 5$ epochs, as it was observed that accuracy tends to fluctuate from one epoch to the next.

As reported in (Table 5.1), biLSTM+ exhibits an improvement over baseline biLSTM in all metrics including a validation accuracy boost of 7-10%, precision boost of 2-7%, recall boost of 3-9% and f1-score improvement of 2-7%. We attribute this performance boost to the ability of biLSTM+ to recognize the pattern of different text sequences which have types of entities following those entities in the input.

BICERE on the other hand shows improvements on accuracy amounting to 2-3% but shows no improvements on validation accuracy which may suggest that more data is needed to improve its performance. It is also noteworthy that BICERE shows an immediate improvement on precision and recall at 100 epochs, but that improvement fades away with higher epochs. The table also shows that loss generally declines as we go from biLSTM to BICERE to biLSTM+ models.

Chapter 6

Conclusion

Our research focused on the design of a virtual assistant for water systems operation. In a field assessment, we implemented physical, artifacts and cultural work models to help us determine the priorities for different features to include in the virtual assistant, affirming the need to support questions about system indicators and inferring the need for personal interaction features, such as small talk and scheduled summaries of the system.

Our technical design focused on the main stages of virtual assistants pipelines; Named Entity Recognition and Intent Classification;

We designed a Named Entity Recognition system that is able to make semantic comparisons to a knowledge base of defined entities in the absence of training data. For this task, we leveraged the representative power of word embeddings by integrating it in the similarity calculations, showcasing significant performance improvements over the baseline TF-IDF and cosine similarity based method. We contrasted the performance for different vector space model representations including NTF, GVSM and directly representing the knowledge base in word embeddings. We also experimented with different types of word embeddings such as FastText and BERT, and different sentence embedding aggregations including averaging and power mean. The most notable factor in the performance improvement was the incorporation of word embeddings as precision boost amounted to 14-20%, recall boost to 6-14% and fl-score boost to 11-17% for word-embedding-based methods over the baseline.

We additionally demonstrated the improvements of integrating information about recognized entities in the input of the intent classification task, which amounted to a 10% accuracy boost, 7% precision boost, 9% recall boost and 7% f1-score boost over the baseline biLSTM model. We also introduced a new classifier architecture (BICERE), which

explicitly encapsulates information about recognized entities which showed small improvement in lower epochs, then little or no improvements over the baseline for higher epochs, which indicates that more data may need to be collected to unlock this model's potential for better classifications.

6.1 Future Work

As shown in the (Section 4.4), the NER task has significantly improved by integrating word embeddings to improve the similarity calculation between parts of queries and curated entities lists. Word embedding models can be further improved by gathering more data related to water operation through periodic web scrapes of online water operation forums, to be aggregated with the current data and retrain the word embedding models. State of the art methods for handling expected misspellings in typed-in text including MOE (Section 4.3.3) should also be contrasted against the currently implemented edit-distance method.

Furthermore, the dataset collected for intent classification is very limited. We recommend implementing an active learning system to query the user about the correctness of an inference, while the current model is in production to collect more data.

We additionally recommend implementing small-talk features to address the needs noted earlier in the field assessment. In advanced stages, question answering features can be integrated as well, providing answers from water systems operations books and learning resources to provide a more complete solution to water operators.

References

[1] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. 2016. 24

[2] Bogdan Babych and Anthony Hartley. Improving machine translation quality with automatic named entity recognition. In *Proceedings of the 7th International EAMT workshop on MT and other Language Technology Tools, Improving MT through other Language Technology Tools: Resources and Tools for Building MT*, pages 1–8. Association for Computational Linguistics, 2003. 15

[3] Surya Bahadur Bam and Tej Bahadur Shahi. Named entity recognition for nepali text using support vector machines. *Intelligent Information Management*, 6(02):21, 2014. 15

[4] Elena Baralis, Luca Cagliero, Saima Jabeen, Alessandro Fiori, and Sajid Shah. Multi-document summarization based on the yago ontology. *Expert Systems with Applications*, 40(17):6976–6984, 2013. 15

[5] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003. 21

[6] Hugh Beyer and Karen Holtzblatt. Contextual design. *interactions*, 6(1):32–42, 1999. 3, 4

[7] Balu Bhasuran, Gurusamy Murugesan, Sabenabanu Abdulkadhar, and Jeyakumar Natarajan. Stacked ensemble combined with fuzzy matching for biomedical named entity recognition of diseases. *Journal of biomedical informatics*, 64:1–9, 2016. 15

[8] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. 23

[9] N. (2018 Canh. Benchmarking intent classification servicesjune 2018. Accessed 2019-09-24, June 2018.

[10] Yufeng Chen, Chengqing Zong, and Keh-Yih Su. A joint model to identify and align bilingual named entities. *Computational linguistics*, 39(2):229–266, 2013. 15

[11] Yukun Chen, Thomas A Lasko, Qiaozhu Mei, Joshua C Denny, and Hua Xu. A study of active learning methods for named entity recognition in clinical text. *Journal of biomedical informatics*, 58:11–18, 2015. 15

[12] K. M. Colby, F. D. Hilf, S. Weber, and H. C. Kraemer. Turing-like indistinguishability tests for the validation of a computer simulation of paranoid processes. *Artificial Intelligence*, 3:199–221, 1972. 8

[13] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008. 22

[14] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017. 25

[15] Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *arXiv preprint arXiv:1805.01070*, 2018. 24

[16] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*, 2016.

[17] O. Davydova. *25 Chatbot Platforms: A Comparative Table*. 2017. Accessed 2019-09-24.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 24

[19] Zixiang Ding, Rui Xia, Jianfei Yu, Xiang Li, and Jian Yang. Densely connected bidirectional lstm with applications to sentence classification. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 278–287. Springer, 2018. 29

[20] Bora Edizel, Aleksandra Piktus, Piotr Bojanowski, Rui Ferreira, Edouard Grave, and Fabrizio Silvestri. Misspelling oblivious word embeddings. *arXiv preprint arXiv:1905.09755*, 2019. 23

[21] Asif Ekbal, Sriparna Saha, and Dhirendra Singh. Active machine learning technique for named entity recognition. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, pages 180–186. ACM, 2012. 15

[22] Christopher Fox. A stop list for general text. In *Acm sigir forum*, volume 24, pages 19–21. ACM, 1989. 17

[23] Richard S Glantz. Shoebox: a personal file handling system for textual data. In *Proceedings of the November 17-19, 1970, fall joint computer conference*, pages 535–545. ACM, 1970. 9

[24] Archana Goyal, Vishal Gupta, and Manish Kumar. Recent named entity recognition and classification techniques: a systematic review. *Computer Science Review*, 29:21–43, 2018. 14

[25] Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. Named entity recognition in query. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 267–274. ACM, 2009. 16

[26] R. Hoffer, T. Kay, and P. Levitan. Smarterchild [natural language software]. ActiveBuddy, Inc, 2001. 8

[27] Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics, 2011. 15

[28] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 29

[29] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016. 29

[30] Holtzblatt Karen and Sandra Jones. Contextual inquiry: A participatory technique for system design. *Participatory design*, pages 177–210, 2017. 4

[31] Mahboob Alam Khalid, Valentin Jijkoun, and Maarten De Rijke. The impact of named entity normalization on information retrieval for question answering. In *European Conference on Information Retrieval*, pages 705–710. Springer, 2008. 15

[32] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015. 24

[33] Adam Lally and Paul Fodor. Natural language processing with prolog in the ibm watson system. *The Association for Logic Programming (ALP) Newsletter*, 2011. 9

[34] Adam Lally and Paul Fodor. Natural language processing with prolog in the ibm watson system. *The Association for Logic Programming (ALP) Newsletter*, 2011.

[35] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016. 15

[36] Yaniv Leviathan and Yossi Matias. Google duplex: An ai system for accomplishing real-world tasks over the phone. Accessed 2019-09-24, May 2018. 9

[37] Xiaohua Liu and Ming Zhou. Two-stage ner for tweets with clustering. *Information Processing & Management*, 49(1):264–273, 2013. 15

[38] Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. Benchmarking natural language understanding services for building conversational agents. *arXiv preprint arXiv:1903.05566*, 2019.

[39] Lajanugen Logeswaran and Honglak Lee. An efficient framework for learning sentence representations. *arXiv preprint arXiv:1803.02893*, 2018. 24

[40] Bruce T Lowerre. The harpy speech recognition system. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE, 1976. 9

[41] Computing Machinery. Computing machinery and intelligence-am turing. *Mind*, 59(236):433, 1950. 8

[42] Deborah L McGuinness, Frank Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, 10(10):2004, 2004. 19

[43] Lian Meng and Minlie Huang. Dialogue intent classification with long short-term memory networks. In *National CCF Conference on Natural Language Processing and Chinese Computing*, pages 42–50. Springer, 2017. 29

[44] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, L Sutskever, and G Zweig. word2vec. *URL https://code. google. com/p/word2vec*, 2013. 22

[45] Marcin Michał Mirończuk and Jarosław Protasiewicz. A recent overview of the state-of-the-art elements of text classification. *Expert Systems with Applications*, 106:36–54, 2018. 28

[46] Diego Mollá, Menno Van Zaanen, Daniel Smith, et al. Named entity recognition for question answering. 2006. 15

[47] Chikashi Nobata, Satoshi Sekine, Hitoshi Isahara, and Ralph Grishman. Summarization system integrated with named entity tagging and ie pattern discovery. In *LREC*, 2002. 15

[48] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. 21, 22

[49] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018. 23

[50] Alexandra Pomares Quimbaya, Alejandro Sierra Múnera, Rafael Andrés González Rivera, Julián Camilo Daza Rodríguez, Oscar Mauricio Muñoz Velandia, Angel Alberto Garcia Peña, and Cyril Labbé. Named entity recognition over electronic health records through a combined dictionary-based approach. *Procedia Computer Science*, 100:55–61, 2016. 14

[51] Khmael Rakm Rahem and Nazlia Omar. Rule-based named entity recognition for drug-related crime news documents. *Journal of Theoretical & Applied Information Technology*, 77(2), 2015. 14

[52] Kashif Riaz. Rule-based named entity recognition in urdu. In *Proceedings of the 2010 named entities workshop*, pages 126–135. Association for Computational Linguistics, 2010. 14

[53] Stephen Robertson. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation*, 60(5):503–520, 2004. 17

[54] Andreas Rücklé, Steffen Eger, Maxime Peyrard, and Iryna Gurevych. Concatenated power mean word embeddings as universal cross-lingual sentence representations. *arXiv preprint arXiv:1803.01400*, 2018. 24

[55] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988. 17

[56] Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. *arXiv preprint arXiv:1805.09843*, 2018. 24

[57] Amit Singhal et al. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001. 20

[58] René Speck and Axel-Cyrille Ngonga Ngomo. Ensemble learning for named entity recognition. In *International semantic web conference*, pages 519–534. Springer, 2014. 15

[59] Rohini Srihari and Wei Li. Information extraction supported question answering. Technical report, CYMFONY NET INC WILLIAMSVILLE NY, 1999. 15

[60] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015. 23

[61] A. Turing. Computer machinery and intelligence. mind. pages 49–1950, 1964.

[62] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 24

[63] Estevan Vilar. Word embedding, neural networks and text classification: what is the state-of-the-art? *Junior Management Science*, 4(1):35–62, 2019. 28

[64] Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339, 1989. 22

[65] Richard Wallace. Artificial linguistic internet computer entity (alice), 1995. 8

[66] Yaqiang Wang, Zhonghua Yu, Li Chen, Yunhui Chen, Yiguang Liu, Xiaoguang Hu, and Yongguang Jiang. Supervised methods for symptom name recognition in free-text clinical records of traditional chinese medicine: an empirical study. *Journal of biomedical informatics*, 47:91–104, 2014. 15

[67] Joseph Weizenbaum et al. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966. 8

[68] Aditya Satrya Wibawa and Ayu Purwarianti. Indonesian named-entity recognition for 15 classes using ensemble supervised learning. *Procedia Computer Science*, 81:221–228, 2016. 15

[69] SK Michael Wong, Wojciech Ziarko, and Patrick CN Wong. Generalized vector spaces model in information retrieval. In *Proceedings of the 8th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 18–25. ACM, 1985. 20

[70] Ledell Yu Wu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes, and Jason Weston. Starspace: Embed all the things! In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[71] Shaodian Zhang and Noémie Elhadad. Unsupervised biomedical named entity recognition: Experiments with clinical and biological texts. *Journal of biomedical informatics*, 46(6):1088–1098, 2013. 15, 21

[72] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015. 29

[73] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*, 2015. 29