ANEC: ARTIFICIAL NAMED ENTITY CLASSIFIER BASED ON BI-LSTM FOR AN AI-BASED BUSINESS ANALYST

Taaniya Arora, Neha Prabhugaonkar,
Ganesh Subramanian and Kathy Leake

Crux Intelligence, New York City, New York, USA

ABSTRACT

Business users across enterprises today rely on reports and dashboards created by IT organizations to understand the dynamics of their business better and get insights into the data. In many cases, these users are underserved and do not possess the technical skillset to query the data source to get the information they need. There is a need for users to access information in the most natural way possible. AI-based Business Analysts are going to change the future of business analytics and business intelligence by providing a natural language interface between the user and data. This natural language interface can understand ambiguous questions from users, the intent and convert the same into a database query. One of the important elements of an AI-based business analyst is to interpret a natural language question. It also requires identification of key business entities within the question and relationship between them to generate insights. The Artificial Named Entity Classifier (ANEC) helps us take a huge step forward in that direction by not only identifying but also classifying entities with the help of the sequence recognising prowess of BiLSTMs.

KEYWORDS

Named Entity Recognition System, Natural Language Processing , Business Analytics, Question Answering Systems, Bi-directional LSTMs

1. INTRODUCTION

At Crux Intelligence, we envisage a break-through in the analytics industry by building an AI based business analyst (ABBA) [1] that performs the functions of a business analyst. The main aim of ABBA is to create a Natural Language interface between the user and the data. This not only helps simplify data access, but also brings the user closer to the data.

Analyza [2], discusses some of the challenges faced while developing such systems. The paper also highlights why Structured Query Language despite being a widely accepted database access tool, is not user friendly and requires far too much knowledge of the physical layout of the database. Thereby it substantiates the use of Natural Language interfaces for such applications.

The most crucial role in such interfaces is played by Question Interpreter which performs the job of understanding user questions and tries to extract structured data from it.

1.1. AI-based Business Analyst

An ABBA supports business leader(s) to make effective decisions. They know the discipline of analytics, understand the data, know how to access and absorb the data and help in decision making. A human business analyst also helps leaders take the right decisions by understanding the business problem, running relevant analysis and producing reports which are easy to consume for the user.

At Crux Intelligence, we are building an ABBA which will help in enhancing the capabilities of a human business analyst and help in making better decisions. Its key component is a question answering system which understands business queries of users and analyses enterprise data to generate appropriate answers. The input to the system is a question entered by a user in natural language. The question is analysed and processed, and the output is an answer, or a list of answers in the form of trends, bar graphs, tables, and numbers. The ABBA is capable of answering the following range of questions:

- **Data retrieval questions:** Direct questions related to entities and metrics. For example, '*What is the sales in New York?'*.
- **Comparative questions:** Questions involving more than two entities, time periods, etc. For example, '*Shipment in Jan vs Feb*', '*East vs West*'.
- **Conditional questions:** Questions having conditions on entities, for example, '*Cities having sales > 3M and < 8M*'.
- **Questions with filters:** Question with filters like Top/Bottom, for example, *'Top 5 stores in Texas'*.
- **Incomplete and non-elucidated questions:** For example, '*Sales*', '*Last Month*'.
- **Questions with complex periods:** For example, '*MTD sales for last 3 years for East*', '*Daily sales from Jan to March 15, 2021*'.

The main task of ABBA is to automatically find the right answer by identifying the entities and intents from the question. Classification of entities is a non-trivial task due to ambiguities present in a user question which may result in classifying an entity into multiple entity categories and hence may lead to different interpretations within the Question Interpreter. We describe this in detail in subsequent sections.

1.2. Named Entity Recognition

Named Entity Recognition (NER) task aims to identify entities in text and classify them into entity categories. It plays a key role in many Natural Language Processing Tasks including Question Answering. Typical examples of entities and entity categories are listed in Table 1.

Table 1. Entity Categories.

Entity Category	Entities
Location	New York, Chicago, Hong Kong
Date	Wednesday, January
Person	Albert Einstein, Mahatma Gandhi
Organization	Google, Tesla, UNESCO

Cuddle.ai [1] describes the role of Question Interpreter in the system and challenges faced during interpretation. One of the tasks within the interpreter is entity extraction, where, for a question, "*How many cars were sold in New York?*", '*cars*' is an entity of category '*Product*' and '*New York*' is an entity of category '*Region*'. It also encounters ambiguities in user questions due to closed domain terminologies where an entity can be classified into multiple entity categories in different contexts. The ability to find correct and relevant answers relies heavily on the Named Entity Recognition task performed on the users' questions.

Several high quality NERs such as those by Stanford, [3] and Spacy, [4] are available. Since these models are targeted at an open domain, they could not be used to meet the special needs of a closed domain system. A major limitation of using such NERs is that they typically classify proper nouns and sometimes numbers or alphanumeric entities like dates as entities. Business entities like '*sales*' or '*number of cars sold*' will remain unrecognized while using such NERs. One of the major reasons of this limitation is that entities in closed domain are not always proper nouns. They can be verbs or even adjectives.

Another approach for identifying entities is by using part of speech tags (POS). In such a scenario, the accuracy of the question interpreter is largely dictated by the accuracy of the POS-Tagger, which is sensitive to case types and the domain on which it was trained. For example, for question, "*What is the sales of Greater Cincy East?*", where '*Greater Cincy East*' is a location. POS Taggers would easily identify '*Greater*', '*Cincy*' and '*East*' as three proper nouns. However, for question, "*What is the sales of greater cincy east?*" we would find that the token '*east*' has been marked as an adjective. Such cases are important to handle in closed domain system where the question is either typed or converted from speech.

Therefore, it is important to have an intelligent domain agnostic model which can also support the specific scenarios discussed earlier. We used BiLSTM architecture to identify entities and classify them into entity categories. The detailed architecture is described in the subsequent sections.

Entity disambiguation becomes even more challenging when the user questions are shorter in length and when an entity gets mapped to multiple categories due to insufficient context in the question. A system to use external knowledge was proposed by Feng [5], where they used a knowledge enhanced Named Entity Disambiguation model which involved using a factual and a conceptual knowledge graph to improve named entity disambiguation for short and noisy texts. We have also used knowledge to further improve our disambiguation performance in the form of custom knowledge dictionary with a different approach which we will describe in section 3.

1.3. LSTMs for Named Entity Recognition

LSTMs [6] are exceptionally capable of learning sequences. Their sequence learning capability find extensive use in NLP. A bidirectional LSTM [7] is even more potent as it makes two passes of the same sequence. Therefore, while tagging an element in a sequence, a BiLSTM not only keeps in mind the past elements but also the elements ahead of it. More advanced neural models have been created for open domain systems using a combination of BiLSTMs with CNNs [8] and CNN along with CRF [9].We chose to use just the BiLSTM model for our named entity classification task as our system deals with a closed domain. The actual meaning of the token has a lower importance in our system in comparison to the sequence it is a part of.

The rest of the paper is organized as follows. We describe various entity categories in section 2, data preparation steps in section 3 and describe the system architecture in section 4. The evaluation procedure is described in Section 5 and the results and error analysis is detailed in Section 6 followed by conclusion.

2. ATTRIBUTES

The process of extraction of structured data from a user question requires us to have certain structured headers under which we categorize the data. We use the term Attribute to refer to an entity identified in a question and attribute class to refer to its category. We will also use these terms in subsequent references. The term Entity signifies a different meaning in ANEC which will be discussed in this section.

The five important attributes considered for named entity classification task are as follows:

- **Entity:** Examples include IDs of *Regions*, *Stores*, *Brands* and actual names like *New York*, *Delhi*, *Texas*.
- **Entity Type:** This refers to the type of entities. For example, *New York* has entity type *City* as well as *State*, *Delhi* has an entity type of *Region*. *Coca Cola* is a brand whereas *Diet Coke* is of type sub-brand.
- **Metric:** A metric is a countable concept as captured in the enterprise database. The derived word '*sold*' corresponds to *Sales* metric.
- **Temporals:** Temporals refer to time and period values. For example, *this week*, *July*, *from Jan 31 2020 to Dec 31 2021* and *January 2018*. It also includes business specific temporals and its abbreviations such as *YTD* (Year to Date), *Q1* (Quarter1), *JFM* (JanFeb-Mar) and *MTD* (Month to Date).
- **Conditions and Filters:** Conditions and filters include words like *highest*, *top-K* and any other conditions that the user wants to apply on the attributes of the question.

3. DATA PREPARATION

The input to the Question Interpreter (QI) is a user question. Two modules within the QI, namely Period Identifier and Condition and Filter identifier, identify Temporals and Condition and Filter attributes from the question respectively. These attributes along with the question are sent as input to ANEC which further identifies and classifies other attributes in the question using knowledge dictionary.

3.1. Knowledge Dictionary Creation

We created three knowledge dictionaries, one each for Entity, Entity Type and Metric attributes. Each dictionary contains words corresponding to its attribute type. Another separate dictionary contains words that occurred across multiple dictionaries. For example, '*Customer Segment Sales*' where '*Customer Segment*' was an Entity Type, '*Customer 01*' an Entity and '*Sales*' as a Metric.

3.2. Data Augmentation

Historical dump of the question database was taken and a total of 1,442 questions were retrieved. The dump consisted of questions, Entity, Entity Type and Metric tokens present in each question. Templating was performed to generate more questions. Each question in the question dump was taken and its attributes were replaced with their respective placeholders. A few complex templates were also generated synthetically. The method used for templating is illustrated in Figure 1.

A few examples of questions generated via templating are illustrated in Table 2. While using the actual IDs in the placeholder for the Entity, the Entity Type was mentioned along with it.

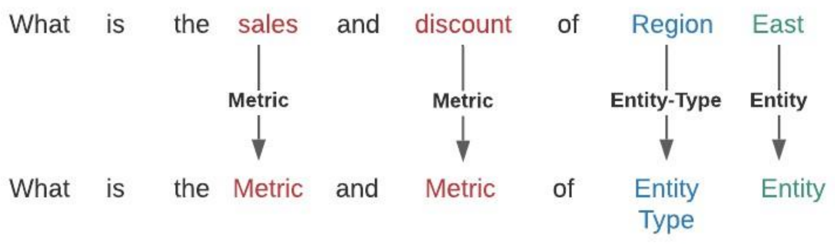
What is the sales and discount of Region East
Metric
Metric
Entity-Type
Entity
What is the Metric and Metric of Entity Type Entity

Figure 1. Templating of questions

Table 2. Replacing placeholders with corresponding attributes

Template	What is the METRIC and METRIC of Entity Type - Entity
Metric 1	Sales
Metric 2	Target
Entity	90
Entity Type	Store
New Question	What is the sales and Target of Store 90
Template	What is the METRIC and METRIC of Entity Type - Entity
Metric 1	Sales
Metric 2	Discount
Entity	East
Entity Type	Region
New Question	What is the Sales and Discount of East
Template	METRIC of (Entity Type – Entity) vs (Entity Type - Entity)
Entity 1	Region West
Entity 2	Region
Entity	9
Entity Type	Region
Metric	Sales
New Question	Sales of Region 9 vs Region West

However, in case of the actual names of entities, the Entity Type placeholder was dropped. The Entity and Metric dictionaries were iterated over, and the placeholders were replaced with tokens from respective dictionaries. In total 11,27,571 questions were generated using templating approach.

4. SYSTEM ARCHITECTURE

The overall architecture of ABBA is represented in Figure 2. In QI, the user question is first passed through Period Identifier, then Condition Identifier and Filter Identifier modules which identify Temporal, Conditions and Filter attributes from it respectively. A detailed architecture of QI including ANEC is illustrated in Figure 3. The question from QI is then tagged with POS tags using Stanford POS tagger [10]. The POS tagged question is converted into a feature matrix which is then sent as an input to the BiLSTM model as highlighted in Figure 4.

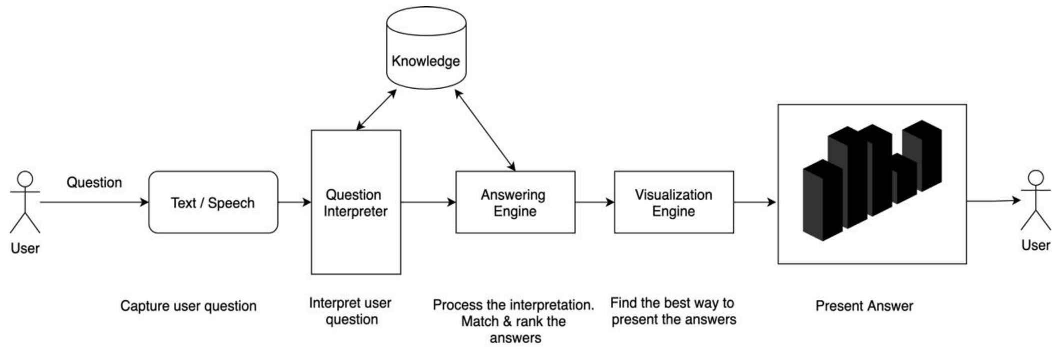

Figure 2. AI-based business analyst

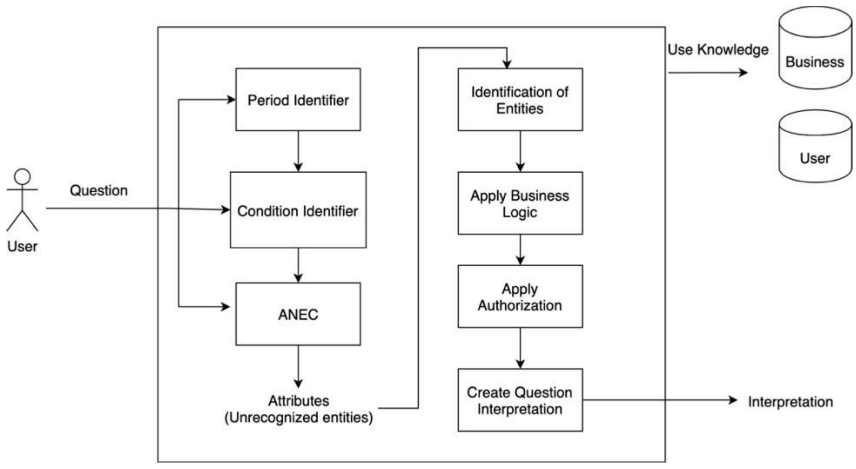

Figure 3. Question Interpreter

4.1. Feature Vector

The default tagset used for Stanford's English POS tagger is Penn Treebank Tagset [11] for POS tagging. These tags were grouped into 8 classes. In addition to these, 4 more classes were defined based on the knowledge dictionary in which each word or its lemma is found in.

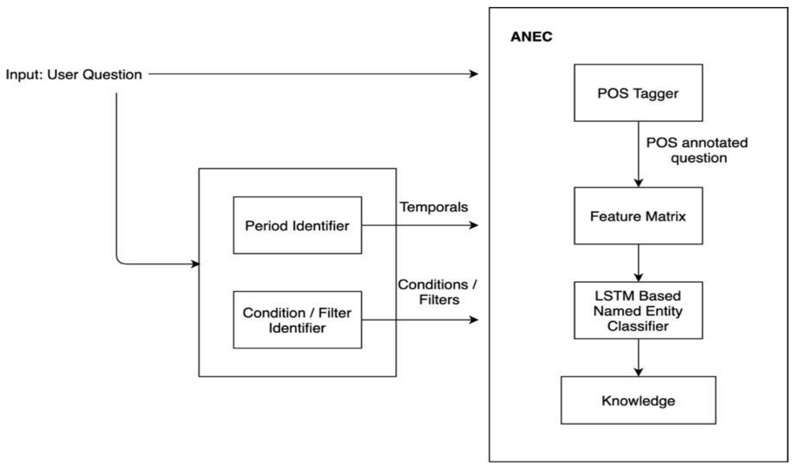

Figure 4. ANEC System Architecture

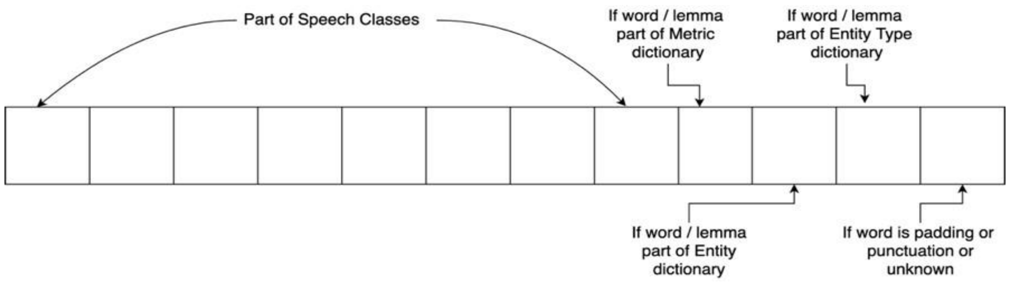
Part of Speech Classes
If word / lemma part of Metric dictionary
If word / lemma part of Entity Type dictionary
If word / lemma part of Entity dictionary
If word is padding or punctuation or unknown

Figure 5. Feature Vector of a word

Classes 9 to 11 are based on the 3 knowledge dictionaries namely - Entity, Entity Type and Metric. Class 12 indicates whether the word is a padding, punctuation or an unknown input. The 12 classes are listed in Table 3 and together they form a feature vector for each word as illustrated in Figure 5.

Table 3. Feature Vector class and their corresponding POS/Dictionary Tags

Feature Vector class	**POS / Dictionary Tag**
Class 1	NNP (Proper noun, singular), NNPS (Proper noun, plural)
Class 2	NN (Noun, singular or mass), NNS (Noun, plural)
Class 3	VB (Verb, base form), VBD (Verb, past tense), VBG (Verb, gerund or present participle), VBN (Verb, past participle), VBP (Verb, non-3rd person singular present), VBZ (Verb, 3rd person singular present)
Class 4	JJ (Adjective), JJR (Adjective, comparative), JJS (Adjective, superlative)
Class 5	CC (Coordinating conjunction), DT (Determiner), EX (Existential *there*), FW (Foreign word), IN (Preposition or subordinating conjunction), PDT (Predeterminer), POS (Possessive ending), PRP (Personal pronoun), RB (Adverb), RBR (Adverb, comparative), RBS (Adverb, superlative), RP (Particle), TO (to) , UH (Interjection)
Class 6	Alphanumeric and CD (Cardinal number)
Class 7	SYM (Symbol), LS (List item marker)
Class 8	WP (Wh-pronoun), WP$ (Possessive wh-pronoun), WRB (Whadverb), MD (Modal), WDT (Wh-determiner)
Class 9	Entity
Class 10	Entity Type
Class 11	Metric
Class 12	Padding, Unknown, Punctuation

4.2. Output

The output for each token from the BiLSTM model is a vector having 6 classes which is reflected in Figure 6. Each of these classes reflect the probability of a word belonging to a particular category with respect to the named entity classification task. The word is tagged with the class having the maximum probability.

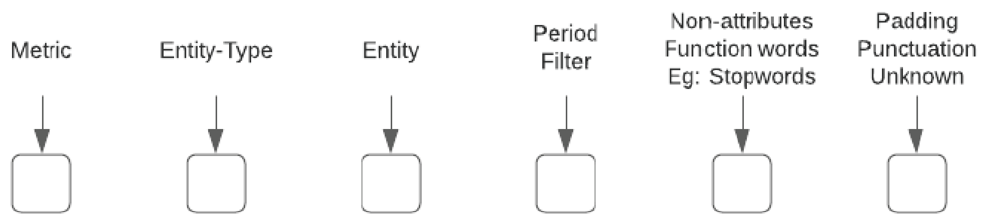
Metric
Entity-Type
Entity
Period Filter
Non-attributes Function words Eg: Stopwords
Padding Punctuation Unknown

Figure 6. Output Vector of a word

4.3. Model Details

A Bi-Directional Recurrent Neural Network with Long Short-Term Memory units is used to predict the named-entity classes from the feature vector. The output of each network for each token passes through a softmax layer to give a probability for each named-entity class. An overview of the flow of data is highlighted in Figure 7.

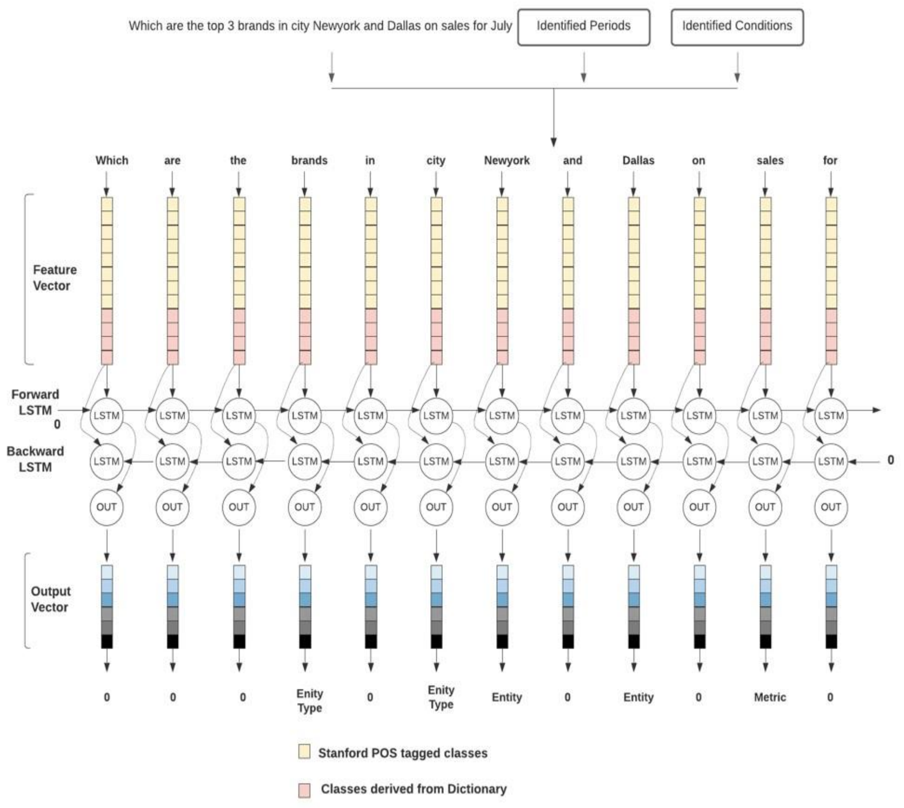

Figure 7. An overview of a question going through BiLSTM Model

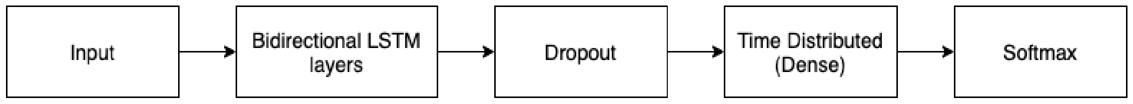

Figure 8. Network Model

The model was implemented in Keras [12] with a TensorFlow [13] backend. The network model is highlighted in Figure 8. The training and hyper-parameters are highlighted in Table 4.

4.4. Knowledge Query

Knowledge Query refers to a query made to the Knowledge Dictionary for linking attribute tokens with their labelled entries in the database.

As mentioned, the output from the BiLSTM model is a vector having 6 classes, each of which reflects the probability of a word belonging to a particular category with respect to the named entity classification task. It is difficult to form a relation between collocated attribute words classified by the BiLSTM model as they have no significant meaning of their own in the absence of any link with the knowledge base. This can be explained with two following scenarios:

Table 4. Model Hyper-Parameters

Parameter	**Value**
Learning rate	0.001
Epoch	40
Batch Size	32
Dropout	0.2
LSTM Units	128
LSTM layers	1

- **Scenario 1:** For multiple consecutive words ‘*United*’, ‘*States*’ and ‘America’, individually identified as attribute of type Entity, the challenge is to determine that the three Entities occur together as a phrase and refer to a nation.
- **Scenario 2:** In some cases, the same word may refer to different attributes in the knowledge dictionary. For example, the word ‘*sales*’ can refer to a metric ‘*Items Sold*’ as well another metric *‘Unit Sales’*. Hence, for disambiguation and to establish a relationship with other attributes, we need to perform a Knowledge Query.

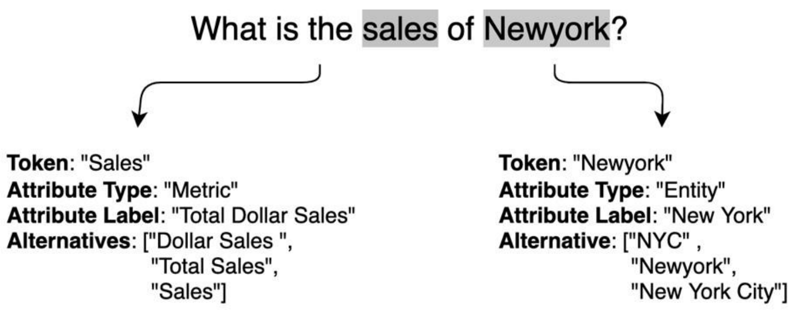
What is the sales of Newyork?
Token: "Sales"
Attribute Type: "Metric"
Attribute Label: "Total Dollar Sales"
Alternatives: ["Dollar Sales ", "Total Sales", "Sales"]
Token: "Newyork"
Attribute Type: "Entity"
Attribute Label: "New York"
Alternative: ["NYC" , "Newyork", "New York City"]

Figure 9. Identifying actual entities in the database from Attribute tokens

An example of the Knowledge Query process is highlighted in Figure 9. The highlighted words refer to the named entities identified by the BiLSTM model as belonging to an attribute class. The 3 steps involved in this process are:

- Grouping
- Disambiguation
- Query

In **grouping**, the tokens belonging to the same attribute category are grouped together as one and successive knowledge queries are performed to extract entity names for the group of tokens. Primary assumptions in grouping process are:

- Successive words labelled with the same attribute class are grouped together. For example, '*Greater*', '*Cincy*', and '*East*' are all entities. If they occur consecutively as '*Greater Cincy East'* in a sentence, they would be grouped together.

- Successive words of the same attribute class, when separated by a single non-named entity, will also be assigned the same group. For example, '*Portland, Oregon*' (Entity) and '*Number of Cars Serviced*' (Metric). This helps account for presence of punctuation marks and stop-words in labels of attributes. One limitation of this assumption is that instances of two distinct attributes occurring together with a coordinating conjunction or a comma might end up being grouped together. For example, '*New York City and Dallas*' and '*New York City, Dallas*'. Here, '*New York City*' and '*Dallas*' refers to the names of two different cities and yet they get grouped together due to this assumption. Such instances are handled by a separate disambiguation algorithm discussed next.

Disambiguation is performed using repeated calls to the Knowledge Dictionary. The Disambiguation Algorithm works as shown in Listing 1. It is also illustrated in Figure 10. The example illustrated is that of the phrase '*Chicago, Dallas, Texas and San Francisco, California*'. The phrase contains three Entities, '*Chicago*', '*Dallas, Texas*' and '*San Francisco, California*' First, all forms of stopwords and punctuation marks are removed. Then, a **query** is made for words starting from the end of the string, one by one into the knowledge dictionary. A successful response means that a match for a particular phrase exists in the Knowledge Dictionary. An empty response indicates that no match was found. In the example, first query is made for the word '*California*'. After receiving a successful response for it, a subsequent query is made for this word along with the word preceding it. Thus, after querying '*Francisco California*' a successful response is received again and next query is made for '*Francisco California*' by preceding it with '*San*' for which a successful response is received as well. Next, on querying '*Texas San Francisco California*' an empty response is received, indicating that this search phrase does not exist in the Knowledge Dictionary. Now, previously stored response of '*San Francisco California*' (highlighted in Magenta) is saved and the remaining words are sent back for disambiguation. The function starts querying again from '*Texas*', and moves on to '*Dallas Texas*' before receiving an empty response at '*Chicago Dallas Texas*'.

Listing 1: Disambiguation Algorithm

~~~
def disambiguate(tokens):
```Disambiguate algorithm for attribute type entity```
#marker: stores position from end of string corresponding to last positive response
#out: stores output response from the present knowledge query marker = 1 result = [] for i in
range(1, len(tokens) + 1):
   out = knowledgeCall(tokens[-i: ], “entity”)    if out != []:        marker = i        results = out if
marker == len(tokens):
   return([[tokens[-marker: ], result]]) else:
   return(disambiguate(tokens[ :-marker]) + [[tokens[-marker: ],
result]] )
~~~

It saves the stored response for '*Dallas Texas*' (highlighted in Red) and moves on to query

‘*Chicago*’. The recursive function finally returns all the valid word groups along with their actual labelled entries in the knowledge dictionary.

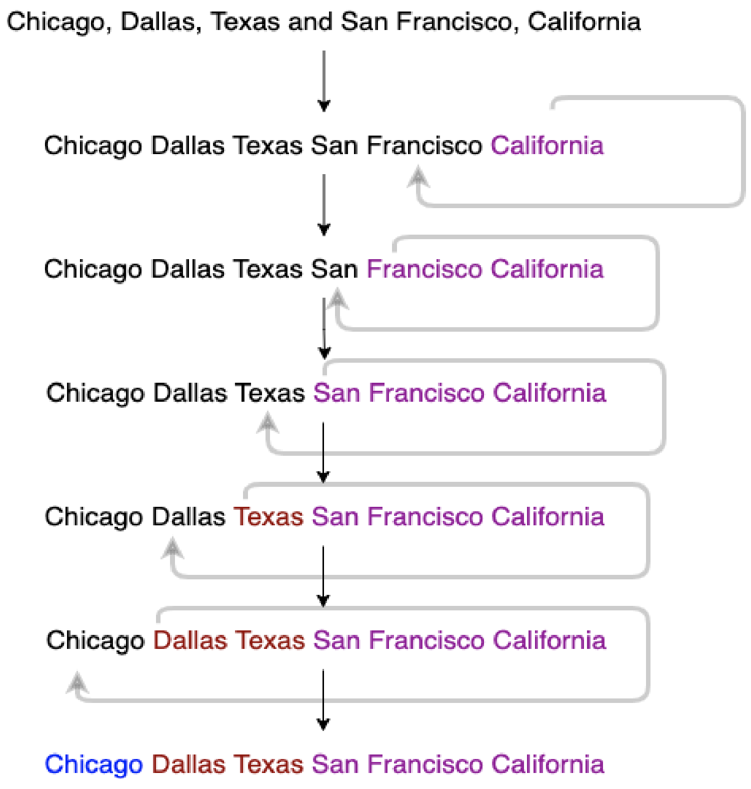
Chicago, Dallas, Texas and San Francisco, California
Chicago Dallas Texas San Francisco California
Chicago Dallas Texas San Francisco California
Chicago Dallas Texas San Francisco California
Chicago Dallas Texas San Francisco California
Chicago Dallas Texas San Francisco California
Chicago Dallas Texas San Francisco California

Figure 10. Illustration of Disambiguation Process

5. EVALUATION

The training, validation and test dataset followed a 60:20:20 split on the questions generated by templating and the system was evaluated on gold dataset of 120 instances with many simple to complex cases created by business analysts. The metrics were calculated at question-level followed by calculation at dataset-level. Each question was evaluated based on the entities (Metric, Entity, Entity Type, Temporal etc) and sibling-relations (Metric-Condition, Filter-Entity Type etc) identified. For each question, the following 3 lists were captured based on MUC evaluation metrics described by [14].

- **Matches:** The entities and sibling-relations that are matched perfectly from both the predicted list and the ground-truth list
- **Spurious:** The entities and sibling-relations that are present in the predicted list, but not in the ground-truth list
- **Missing:** The entities and sibling-relations that are present in the ground-truth list, but not in the predicted list

The evaluation is done using the following 3 metrics using the above lists:

- Precision: |Matches|/(|Matches| + |Spurious|)
- Recall: |Matches|/(|Matches| + |Missing|)

- F1-score: $\frac{2 \times Precision \times Recall}{Precision + Recall}$

The dataset-level metrics are computed from aggregating question-level metrics by computing their micro-averages and finally the accuracy for the dataset is computed as follows:

$$Accuracy = \sum_i \frac{I_A F_i}{n}, \text{ where}$$

1. F(i) is the F1-score of the ith interpretation
2. I(A) is an indicator function that returns 1 if F(i) = 1, else 0
3. n is the total number of samples in the dataset.

Table 5. Performance scores on Gold Dataset

Metric	**Partial Comparison**	**Strict Comparison**
Precision	0.979	0.979
Recall	0.979	0.979
F1-score	0.979	0.979
Accuracy	0.987	0.987

The evaluation is done based of 2 types of comparisons between the predictions and ground truth. The 2 comparators used for this purpose are:

- **Strict comparator:** All properties of the entities and sibling-relations must be equal for two entities and sibling-relations to be considered equal.
- **Partial comparator:** Even if the span (start and end indices) of an entity or siblingrelation is incorrect, as long as the other properties of the entity or sibling-relation is correct, they are considered to be equal.

The accuracy was measured in terms of number of questions with all the attributes classified correctly. Instances of questions where entities were classified partially, were treated as an incorrect classification. The result of the experiment is reported in Table 5.

We have not published the performance of other popular Named Entity Recognizers in comparison against our system. Firstly, this is because ANEC was built to classify attributes in a closed business domain whereas other NERs were built for more general open domain tasks. Secondly, by classifying attributes with ANEC helps us save a large number of calls to the Knowledge Dictionary. In case of a standard NER, we have to make a large number calls to the Knowledge Dictionary just to determine which class of Attributes a phrase belongs to. Hence, comparing two systems aimed at different domains would not be a fair comparison and the results would be highly biased towards ANEC.

6. RESULTS AND ANALYSIS

We have divided the results in the following three categories:

- Attributes present in single knowledge dictionary
- Attributes present in multiple knowledge dictionaries
- Attributes not present in any knowledge dictionary

Following subsections present the results and examples from each category.

6.1. Attributes Present in Single Knowledge Dictionary

The system was able to identify and classify all attributes which were present in a single knowledge dictionary. For example - *Sales* and *Sales Achievement* were present only in the Metric dictionary. Similarly, words like *Region* and *Store* were present only in Entity Type knowledge dictionary.

6.2. Attributes Present in Multiple Knowledge Dictionaries

For attributes present in multiple dictionaries, the system was able to classify attributes correctly when the attribute word was present with other words as an attribute. For example, the word *Customer* was present in the Entity Type dictionary as *Customer Segment*, whereas it was present in the Entity dictionary as *Customer 01*. For ambiguous cases like *Customer*, it was easy to both identify as well as classify them with the help of adjacent words.

In case of words wholly occurring in multiple dictionaries, it becomes difficult to classify them. For example, *Target* occurs by itself, both as a Metric as well as an Entity. Another example is the word *Store*, which is used both as a Metric as well as an Entity Type. In such cases, the system is able to identify named attributes but is unable to classify them with sufficient confidence. A few such ambiguous questions are highlighted below.

1. What is the **Target** and **Sales** for West Region?
2. What is the **Sales** for **Target** for West Region?

In the first example, the word *Target* is a Metric along with the word *Sales*. In case of second example, the word *Target* is an Entity whereas the word *Sales* remains a Metric. In such cases, our system fails to classify the target with sufficient confidence.

6.3. Attributes not Present in Any Knowledge Dictionary

For words which do not occur in any of the attribute dictionaries, it is crucial that we identify them even though there is no way to classify them correctly. A few examples of such tokens are *Performance* and *Productivity* which were not present in any knowledge dictionary but had some significance in the business sense. Our system was able to identify them, but, since they were not found in the knowledge, they were marked as unrecognized attributes.

7. CONCLUSION

In this paper, we utilize our Recurrent Neural Network with BiLSTM units to identify and classify named entities in natural language questions. We have also provided an overview of the techniques employed to develop a Neural Network based NER in context of an AI Based

Business Analyst. Availability of a large collection of annotated data was very important to train a deep learning model, so this paper also discussed about templating approach which was used to create a large training dataset from a small sample set of 1,442 questions.

While our BiLSTM model is effective in identifying and classifying a large majority of questions, it falls a tad bit short when it comes to identifying context in very complex cases as highlighted in section 6.2. The resolution of such ambiguities needs extensive research into business attributes and how they are linked together by various stop-words and function-words. An extension of the ANEC pipeline can be inclusion of a spell-checker. Attributes with spelling mistakes usually get added to the list of unrecognized entities. A spell-checker module can help us identify attributes present in the knowledge dictionary from the list of unrecognized attributes. Another addition can be usage of word embeddings as features for better classification of attributes that are not present in any knowledge dictionary.

ACKNOWLEDGEMENTS

We would like to thank all the members of Crux Intelligence for their support and contribution. We would also like to thank Samriddhi Sinha for his valuable contribution in building the system.

REFERENCES

[1] Joshi, A.; Prabhugaonkar, N.; Hadaye, R.; Unnikrishnan, S.; Das, S.; Gala, N.; Bari, P.; and Mall, N. 2019. 'I Under- stand What You Asked': Question Interpreter for an AI- Based Business Analyst. In Arai, K.; Kapoor, S.; and Bhatia, R., eds., Intelligent Systems and Applications, 1282–1288. Cham: Springer International Publishing. ISBN 978-3-030- 01057-7.

[2] Dhamdhere, K.; McCurley, K.; Nahmias, R.; Sundararajan, M.; and Yan, Q. 2017. Analyza: Exploring Data with Conversation. *Proceedings of the 22nd International Conference on Intelligent User Interfaces.*

[3] Finkel, J. R.; Grenager, T.; and Manning, C. 2005. Incorporating Non-Local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, 363–370. USA: Association for Computational Linguistics.

[4] Honnibal, M.; and Montani, I. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

[5] Feng, Z.; Wang, Q.; Jiang, W.; Lyu, Y.; and Zhu, Y. 2020. Knowledge-Enhanced Named Entity Disambiguation for Short Text. In *Proceedings of the 1st Conference of the Asia- Pacific Chapter of the Association for Computational Lin- guistics and the 10th International Joint Conference on Nat- ural Language Processing*, 735–744. Suzhou, China: Asso- ciation for Computational Linguistics.

[6] Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation*, 9: 1735–1780.

[7] Schuster, M.; and Paliwal, K. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.*, 45: 2673– 2681.

[8] Chiu, J. P. C.; and Nichols, E. 2016. Named Entity Recognition with Bidirectional LSTM CNNs. *Transactions of the Association for Computational Linguistics*, 4: 357–370.

[9] Ma, X.; and Hovy, E. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1064–1074. Berlin, Germany: Association for Computational Linguistics.

[10] Manning, C.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S.; and McClosky, D. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 55–60. Baltimore, Maryland: Association for Computational Linguistics

[11] Marcus, M.; Kim, G.; Marcinkiewicz, M. A.; MacIntyre, R.; Bies, A.; Ferguson, M.; Katz, K.; and Schasberger, B. 1994. The Penn Treebank: Annotating Predicate Argument Structure. In*Proceedings of the Workshop on Human Language Technology*, HLT '94, 114–119. USA: Association for Computational Linguistics. ISBN 1558603573.

[12] Chollet, F.; et al. 2015. Keras. https://github.com/fchollet/ keras.

[13] Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Leven- berg, J.; Mane´, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Tal- war,K.;Tucker,P.;Vanhoucke,V.;Vasudevan,V.;Vie´gas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; and Zheng, X. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org.

[14] Chinchor, N.; and Sundheim, B. 1993. MUC-5 Evalua- tion Metrics. In *Fifth Message Understanding Conference (MUC-5): Proceedings of a Conference Held in Baltimore, Maryland, August 25-27, 1993.*