International Journal on Natural Language Computing (IJNLC) Vol.10, No.4, August 2021

DOMAIN BASED CHUNKING

Nilamadhaba Mohapatra, Namrata Sarraf , Swapna sarit Sahu

Department of Data Science, Zeotap, Bangalore, India

ABSTRACT

Chunking means splitting the sentences into tokens and then grouping them in a meaningful way. When it comes to high-performance chunking systems, transformer models have proved to be the state of the art benchmarks. To perform chunking as a task it requires a large-scale high quality annotated corpus where each token is attached with a particular tag similar as that of Named Entity Recognition Tasks. Later these tags are used in conjunction with pointer frameworks to find the final chunk. To solve this for a specific domain problem, it becomes a highly costly affair in terms of time and resources to manually annotate and produce a large-high-quality training set. When the domain is specific and diverse, then cold starting becomes even more difficult because of the expected large number of manually annotated queries to cover all aspects. To overcome the problem, we applied a grammar-based text generation mechanism where instead of annotating a sentence we annotate using grammar templates. We defined various templates corresponding to different grammar rules. To create a sentence we used these templates along with the rules where symbol or terminal values were chosen from the domain data catalog. It helped us to create a large number of annotated queries. These annotated queries were used for training the machine learning model using an ensemble transformer-based deep neural network model [24.] We found that grammar-based annotation was useful to solve domain-based chunks in input query sentences without any manual annotation where it was found to achieve a classification F1 score of 96.97% in classifying the tokens for the out of template queries.

KEYWORDS

Natural Language Processing- Named Entity Recognition, Chunking, Recurrent Neural networks, Transformer Model, Automated Annotation, Transfer learning, Grammar based sentence generation

1. INTRODUCTION

Chunking is the process of splitting the words of a sentence into tokens and then grouping these tokens in a meaningful way. These chunks are used to solve our relevant NLP tasks[3]. It labels every word of the sentence suitably and thus lays out a basic framework for recognition of larger words which will be used for tasks such as question answering, information extraction, topic modeling, NER, etc[16]. Named Entity Recognition as described in the papers [2][10][11]is the process of extracting and tagging words that denote the names of certain places, people, organizations, time, etc. Several Natural Language Understanding Tasks like POS tagging, Tokenization, and Noun Phrase Identification are all chunking tasks.

Our task for the specific domain(advertising and marketing segment creation) was to parse the input search query (in our use case it represents the client’s expected marketing theme), and prefilter different attributes corresponding to the input sentence for programmatic segment creation, i.e. automated user segment creation by filtering user profiles with respect to attribute values retrieved after parsing client advertising briefs, used for online targeting. This input sentence is nothing but an advertising or marketing brief which is a summarised description provided by advertisers of their target audience identified using online identifiers. The starting point of solving this task is to do some rule-based parsing such as finding meaningful n-grams tokens based on Part of Speech tags and then searching corresponding matches in the domain data catalog.

The data catalog mainly consists of all profile attributes, for an online identifier such as MAID, cookies. This contains the Interest, Travel history, Demographics, APP installs, etc. Further, these categories have more granular subcategories and the final sub-subcategories. These attributes are considered phrases when used for matching the incoming n-grams of the input client brief. The initial finding suggested that all the tokens in the input query or input client brief may not be relevant for matching the corresponding data catalog entries. This gives rise to the conceptualization of segregating the search query into substructures of context and phrases. These contexts and phrases are nothing but chunk grams. To convert this problem to a machine learning task we used custom SIO(inside, outside, start) tags for phrases and context separately. So each token in the client brief would be associated with a custom chunk tag (phrase-inside, phrase-outside, phrase-starting,context-inside, context-outside, context-starting). Later these tags are used in conjunction with pointers frameworks to find the final chunk.

Finally, the task becomes a classification-based ML problem where for any given input we have to predict custom chunk tags for each token. But to build this machine learning system we need an annotated corpus to train the model. Chunking has also previously been used in various domain-level projects with the end goal of retrieving custom substructures of a sentence[18] but solving a chunking task for a very specific domain is tedious. The chunk meanings are domain-specific and we require large-scale, high-quality annotated corpus. Due to a lack of aforementioned, we have used a grammar-based sentence generation approach to create a data set for training similar to [25]. Where systemic functional grammar(SFL) is used to solve the problem of sentence generation, here SFL brings together attention to inter-relationships, linking alternative forms of expression. In our language generation approach every query is labeled with our custom chunk tags. As a result, we generated a large number of annotated sentences using a few production rules defined under our grammar formalism. These sentences were used as a data set for training our classification model.

To generate labelled training sentences, we start out with basic template or grammar definitions correlated with inferences drawn from real world search query examples. Our grammar follows a modified version of the context-sensitive grammar[26] where we define the set of terminal and non-terminal nodes and a finite set of production rules. Unlike Chomsky’s Natural Form(CNF) or Greibach normal Form of grammar[28], our defined grammar is unrestricted in the definition of its production rules, There are however certain added features like repetitive or redundant substitutions and mandatory substitution rules which are applied for custom entity-tag(chunk tag) to carry out the annotation of every token of the query generated using the grammar. This approach minimizes the role of the lexicon of a word for chunking i.e. 'chunk grams' and introduces custom NER tags to build the same.

Once we have generated the training data the task reduces to using the data through an efficient architecture and building a relevant machine learning model. We have used an ensemble based deep neural network[24] model which consists of a fine-tuned Transformer Model and a recurrent neural network model with explicit pos tags together to predict tags and chunk substructures of a sentence[24]. The ensemble model consists of two different architectures. The first one is a transformer model. The Transformer Model[9] helps in attention maximization and hence expanding the learning abilities of the model. It provides embeddings for words in a sentence highly correlated with other words in the sentence owing to its multi-head attention mechanism[19], and thus reflects a more relative semantic of a word instead of using just the POS tag for embeddings.[9] We have used pre-trained models that facilitate the understanding of words and their place from a much larger dataset and fine-tuning it further customizes and improves the model understanding of those words in the specific domain and its corresponding tags associated with it. The second head of the ensemble framework is a RNN model. In our Recurrent Neural Network section of the Model, we use Bidirectional LSTM in addition to Convolutional Neural Networks[8]. We introduced this segment of the model to compensate for the shortcomings of the Transformer Model and give a more general approach to Tagging and segmentation for chunking. Since LSTMs use the sequential word by word approach of processing, they can be slower in processing, and hence we use simpler shorter custom embeddings targeting exactly the common error points of Transformer Models[9]. The word embedding essentially utilizes a combination of positional embedding, POS Tag, and the word vector generated after whitespace tokenization. Our analysis of the Roberta Model[9] showed an error in differentiating similarly Pos tagged tokens and hence a boost to the Part of speech tagging is given here. The BiLSTM network is complemented with a stack of CNN layers[8] to aid feature extraction from sentences and the feedback loop helps in accurately labeling these features and further segmenting them into chunks.

2. PRIOR WORK

In the beginning we studied Named Entity recognition with custom entity labels, i.e. IOB annotated context phrase tags and further chunking using these IOB notations. The NER task is performed as a token classification problem using deep neural network models, and a pointer framework is used to employ the IOB notations hence produced to extract chunks of context and phrases out of the query. Entity Recognition was initially performed using extensive knowledge base systems, its orthographic features, ontological and lexicon rules[4][2][14][15]. However, the new trend has shifted towards neural network-based structures to define entity relations [5][13]. Hence the mentioned top 20 state-of-the-art NER mechanisms are neural network-based including LSTM, GRUs, BERT, CNN, and a combination of them as suited. [2] [6][10][11][12]. Chunking has been done using machine learning-based models such as HMM(Hidden Markov Model) [7][17] and Maximum Entropy model and has gradually seen a shift to Statistical models such as Support Vector Machines and Boosting [8], [3], [7]. In more recent times, Neural Models have been on a rise as a tool for chunking. Neural network models are deployed as a classification system to classify Beginning, Inside, and Outside of the chunk tags required or also known as BIO tagging which is quite a popular Named Entity Recognition mechanism for segmentation. The latest paper submitted by IBM Watson uses a combination of Bi-LSTM and CNN to label the tokens of the sentence and then chunk them together accordingly [8]. They follow an encoder-decoder-pointer framework while segmenting and labelling chunks sequentially using a pointer. We use POS tag reinforced word vectors as input to this segment of the model. Since the model excelles majorly in sequential labelling and feature extraction, it helps lessen the gap between similar tags occurring together frequently. The best suited and well performing architecture that we used in our paper is an ensemble model[24] using the Transformer-based Model RoBERTa [9] and recurrent neural network[8] for labeling and segmentation, after which we group the hence labeled chunks and map the contexts and phrases together.

Domain Based Chunking refers to chunking of user specific sentences for example search queries, advertiser briefings, etc into custom entity chunks. This therefore requires an abundance of accurately labelled data for optimised neural network model training. The data should exhibit diversity and should be a collection of meaningful phrases or sentences relevant to the specific use case. For this purpose we have deployed our own grammar formalism for the query generation task which refers to our domain specific data catalog for vocabulary.

Sentence generation using grammar rules such as CNF, CSG[26] or GNF[28] has been worked upon for quite some time[29]. However there is little to few literature which touch upon a more generic data generation task for extensive deep learning purposes, and further less for annotated data or well structured sentence generation for NLP tasks such as Chunking, NER etc. While CSG[26] is an unrestricted form of grammar, CNF and GNF[28] specify definitive production rules which are quite helpful in tracing back the initial structure of the grammar. CNF requires at the minimum two non terminal nodes to be used as substitutions at least once, whereas GNF puts no such restrictions on non terminal nodes being used or not used for substitutions in production rules, it however restrict the use of stop symbol ε to the Start symbol only. More evolved neural network based works range from using language Models based over LSTM [31] to using more rule based systems over Context free Grammar. The main component for efficient Text generation is optimised Grammar formalism, which should be easily parsable and as generalised as it could be to generate a wide range of sentences covering most of the vocabulary. The vocabulary being referenced can be specific to the domain of the use case as in our experiment or it can be a more general corpus, for example EnglishWikipedia[23] or BOOKcorpus[32]. Attention has also been used in recent advancements in the field of text generation where [30] uses Multi Modal Attention Branch Networks. This paper uses visual scenes as input to generate natural sentences for domestic robots.Their approach uses a linguistic attention branch mechanism in conjunction with several other attention branch mechanisms to generate instructions. System for Natural Language Sentence Generation VINCI is a program which takes formal description as input of any language, and produces strings in the language.

However in this task we not only need well semantically structured sentences but Annotated structured sentences. Annotated means that there should be certain tags or inferences associated with tokens which can then be used as labels for training data as is used in multiple Natural Language Inference Tasks. Language generation, while being a part of wide range applications such as Chatbot[33][34] and text summarization[35], is not enough to generate labelled data which is highly costly to carry out manually. Since Deep learning requires quite a lot of accurately labelled data, our grammar formalism attempts to serve this use case. We use custom tags and vocabulary for domain level labelled text generation. The query or sentence is generated word by word until all variables of a single production rule have been traversed. To widen the range of sentences we can add different Production rules or delete similar ones. The mandatory primary substitutions for the variables are the tags to be associated with the token and the sum union of tags used in all production rules can also be customised as per the requirement of the use case. The classification F1 score achieved ranges from 90% to 96.97% as we scale the data from 6000 training and 2000 testing queries to 144000 training and 24000 testing queries. Refer table 3.

3. EXPLAINING THE DOMAIN SPECIFIC TASK

This problem statement is inspired while building a domain-specific search engine i.e. to understand the input queries we need a domain-specific parser. From our analysis, we found out that from the whole query sentences we were interested only in certain types of chunks grams. We call them context and phrases where each phrase may be associated with a context. Now the task becomes; given a query sentence, to find the context and phrase pairs.

For example Young men using Facebook and Netflix, traveling to Delhi
After parsing: {“NA”: “young men”, “using”:[“Facebook”,” Netflix”],” traveling to”:[Delhi]}

Note: one context may have multiple context phrases and some context phrases might not have any specific context.

Here context shapes the subspace of the search concerning which categories and which subcategory groups need to be explored, whereas phrases represent the value that needs to be matched and retrieved from inside those corresponding categories or subcategory groups in the data catalog.

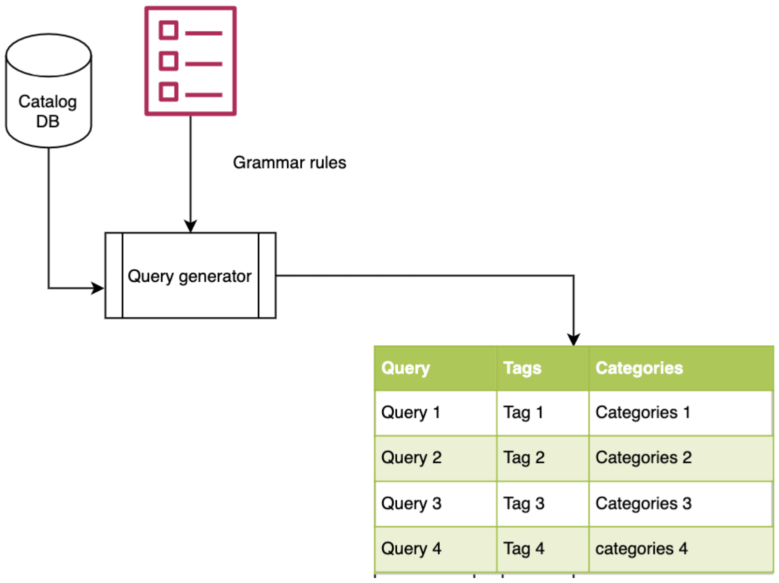
From a business perspective, a search query elucidates in our use case is the client's brief. It could be a well-formed sentence or a short phrase, or a group of attributes separated using commas. As a result, the structure of the query could range from highly structured to very unstructured sentence format. Hence applying a rule-based mechanism to the entire range of expectant queries, and anticipating high accuracy becomes a difficult task. This brings in the need to deploy an ML-based algorithm to chunk them and group them into contexts and phrases.The data catalog we reference for sentence generation is native to our use case, wherein it represents the topology of user attributes that are used for programmatic targeting of users. It is of a flattened tree-type structure, where the higher-level categories(example: Apps used) are divided into subcategories(example: Gaming Apps, Entertainment Apps, Social Apps) and sub-subcategories(example: Sudoku, Netflix, Instagram, etc.) of attribute values. These attributes further help to segment the user profiles required for the client's advertising audience.

Contrary to the rule based system where we split the words using whitespace tokenizers, and used Part of Speech tags to chunk together these tokens into context and context-phrases, in this paper we replaced these POS tags in our ML-based system with custom entity labels or SIO(Start-Inside-Out) Context Phrase tags. These custom chunk tags along with the pointer framework are used to retrieve the context and phrases from the input sentence. But to train the machine learning system we needed annotated data with these custom chunk tags. The data generation part is explained in the following data section.

4. DATASET PREPARATION

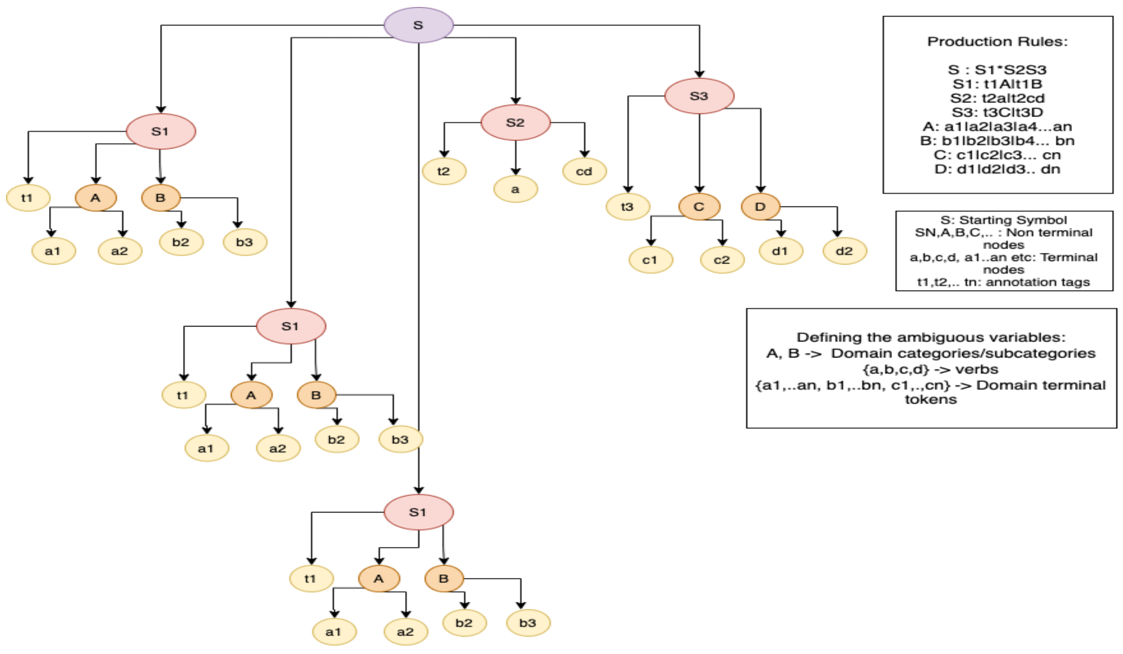
The above classification task requires an annotated training corpora to train our machine learning based ensemble deep neural network model[24]. Although we are using transfer learning[9] for cold starting our problems, we still need a domain specific diverse set of annotated training corpora. In our experiment we try to automate the annotation process as a sentence generation task Refer Figure 1. For this purpose we define a modified version of a context-sensitive form of grammar. The defined Production rules specify a tag to be associated with every token being augmented to the final sentence. These tags are SIO labelled Context and Phrase Tags and an Other tag represents the supporting tokens to maintain structural semantics of the sentence. Repetitive substitution and multiple length token substitution of non terminal nodes are also employed to produce sentences as similar to expectant queries as possible.

We have used two distinct substructure tags to identify contexts(CXT) and phrases(PHR) out of a query and these two tags are SIO annotated. With a defined json file of grammar, refer Figure 2 for the derivation tree, we generated the data (query sentences) to train our model using attribute values retrieved from the data catalog as the terminal substitutions.



Query	Tags	Categories
Query 1	Tag 1	Categories 1
Query 2	Tag 2	Categories 2
Query 3	Tag 3	Categories 3
Query 4	Tag 4	categories 4

Figure 1: Data generator: Set of predefined grammar rules are fed into the query generator to output annotated query and their tags



Production Rules:

S : S1*S2S3
S1: t1Alt1B
S2: t2alt2cd
S3: t3Clt3D
A: a1la2la3la4...an
B: b1lb2lb3lb4... bn
C: c1lc2lc3... cn
D: d1ld2ld3.. dn

S: Starting Symbol
SN,A,B,C,.. : Non terminal nodes
a,b,c,d, a1..an etc: Terminal nodes
t1,t2,.. tn: annotation tags

Defining the ambiguous variables:
A, B -> Domain categories/subcategories
{a,b,c,d} -> verbs
{a1,..an, b1,..bn, c1,.,cn} -> Domain terminal tokens

Figure 2: Derivation Tree for the Grammar

A production rule defines the structure of the queries to be generated, containing variables in a defined sequence. Refer Figure 3, The variables are then specified with a list of their associated candidate substitutions. The candidate substitutions can be either of the three;

1. Terminal Substitutions:

    a. Context token: example: “using”, “travelling” , “living” etc are used as terminal substitutions annotated to Context tags

b. Other tokens: examples: “a”,”an” , “the”, “to” , “from” etc are used to provide a meaningful structure to the query generated.
c. Phrase tokens: These are sub subcategory attribute value retrieved from domain specific data catalog .These substitutions can be single length tokens(“Netflix”,”Facebook”,”Male”) or multiple token phrases(“Disney Hotstar” “Amazon Prime Video”, “United States of America”). Since we directly fetch tokens or terminals from the data catalog containing the attribute data, we need to pre-process it before it can be used as substitution while query generation.

    i. Longer app names are processed and only their keywords are used as candidate substitutes for the variables defined in the grammar.
    ii. Synonyms of attribute values are also used to diversify the range of candidate tokens getting augmented to the final generated queries.

2. Non terminal Substitutions: The non terminal substitutions constitutes the category of attribute value(example: “Demographic”,”Apps Installed”, “Intent”) or the subcategory of attribute value(“Apps.Social”,”Demographic.Location”, “Apps.Game”) . A * mark follows these non terminal substitutions to suggest all their sub subcategory attribute values to be used as candidates for terminal substitutions.

    a. For Example, the non terminal: “Interest.*” denotes that all the sub subcategory values(“Dance”,”Music”,”Games”, etc) available for category Interest in the data catalog will act as candidates for the terminal substitutions. of “Interest.*”

3. Custom entity tag to annotate the terminal substitutions {PHR_S/I/O, CXT_S/I/O, O,DIFF}. The Custom entity tag is mentioned as the first substitution for every variable, Refer Figure 3. SIO tagging is carried out for multiple length terminal attribute value substitutions

A level of diversity is maintained when defining different production rules, so that the sentences generated encapsulates short phrases (example: “young men”), comma separated attributes (example: “Netflix, Facebook, Instagram gamer”) and Well structured sentence format (example: “Men aged 35 using Netflix and Facebook”).

```
 {
"Production Rule":"S1_S2_S3_S4",
"S1":["PHR_S/I/O","Demographics.Gender.Male","Demographics.Gender.Female","men","
women","girls","boys","children","child","man","woman","people"],
"S2":["CXT_S/I/O","planning","looking","intending","planned","intended","interested","inter
esting","like","likes","liking","love","loves","loving","playing","enjoying","enjoy"],
"S3":["O","to","in","a","an","the"],
"S4":["PHR_S/I/O","Intent.*"]
 }
```

Figure 3: Example Production Rule defined in Grammar

We also use repetitive grammar after drawing inferences from real world input queries, where multiple sub subcategory attribute values from the same category or subcategory may be present in repeated conjunction in a single search query. Repetitive Grammar (Refer Figure 4) is assigned the specific ratio for single, double, triple or as mentioned number of repetitions for any candidate terminal token. In Figure 4 all sub subcategory attributes present for category Apps in the data catalog are taken as candidates for repetitive substitution. The main purpose of repetitive substitution is to generate queries as close to the expectant queries as possible. It gives us control over the number of repetitions allowed and thus enhances the quality of generated queries.

```
{
"Production Rule":"S1.*.S2",
"S1.*.S2":["PHR_S/I/O","Apps.*"],
"S2":["O","and",",","or","with"]
}
```

Figure 4: Example Production Rule with Repetitive Grammar

8 different tags {PHRS, PHRI, PHR0- SIO annotated Context-Phrase, CXTS, CXTI, CXTO-SIO annotated Context, DIFF, O- Supporting tags to generate meaningful sentences(example: a, an, the , to, from)} are used to annotate each token of the query generated. This helps us primarily in annotating the data. When a terminal substitution has multiple tokens, tags PHR/CXT-I and PHR/CXT-O tags come into play where PHR/CXT-I symbolises Inside of Phrase/Context and PHR/CXT-O symbolises Outside of Phrase/Context. These tags define if the token is a context(CXT-S/I/O) for the query or a phrase(PHR-S/I/O). The 'O' (Other) tag is used to define tokens which do not add much to the semantics of the query but are important for structure meaningfulness of the query and the 'DIFF' tag is a special tag used to define tokens categorized as an otherwise Other tag will be used as Context , this is done to override confusion.

The number of production rules used for our experiment are 22. Every Production rule generates a specific number of queries which is mentioned while running the script designed for it Refer Figure 1. Since we are using self generated data referencing our domain specific database, we can tweak the amount of relevant data we need. The size of data on which the present model is trained upon is 144000 training sentences and 240000 testing sentences, which sums up to 28MB of uncompressed text. The huge number of training and testing sentences is due to the variability and diversity of the production rules defined. The fine tuning is done on a model pre-trained on 160GB of uncompressed text.

5. EXPERIMENTS

We broke the experimentation down into two processes. The first process was to benchmark a novel ensemble deep neural network based classification model for chunking. The BIO annotated POS chunk tags were used as token classification labels and the model was validated on public data ConLL2000 [21]. The Second process is using the ensemble deep neural network framework and utilising Transfer learning we fit the custom chunk tags(SIO annotated context phrase tags) on the generated dataset and analyse the performance.

5.1. Ensemble Deep Neural Model for Classification (TASK-1)

The purpose of ensemble training is to boost the learning capabilities of neural architecture. We deploy this model to further increase our F1 score by using two complementary training models. RoBERTa[9] uses its encoder to tokenize and embed the tokens into feature vectors. In the Bi-LSTM+CNN[8] model, we use a custom tokenizer after analysing the misclassifications of RoBERTa[9].

5.1.1. RNN based Structure

The BiLSTM and CNN[8] Refer Figure 1, approach takes in custom embeddings built using one hot encoded Part of speech , word vector and relative position of the token in the sentence, Refer Equation 1. As RNNs are used to efficiently capture sequential information of the text, LSTMs give an edge of handling exploding and vanishing gradient problems, and enables a more long memory application. Since LSTMs have the ability to encode context information but don't treat each chunk as a complete unit, CNN and max-pooling layers are employed to extract features from chunks. For each identified chunk, we first apply CNN to the embedding of its words (irrespective of it being a single-word chunk or chunk), and then use the max-pooling layer on top to get the chunk feature vector for labeling. The above mentioned BiLSTM+CNN[8] framework however suffers from the Limited Memory problem and fails to capture context in a long sentence.

word embedding = positional encoding*([word vector, ohe pos tag]) ...Eqn.1

where,

positional encoding = position of token in query / length of query
ohe pos tag = one hot encoded part of speech tag.

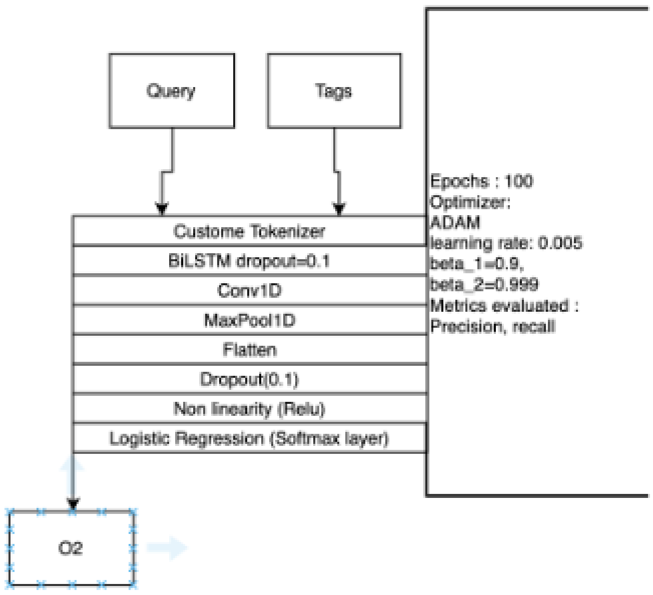


Figure 5. BiLSTM+CNN: This model produces Output 2 or O2 after the query goes through a custom tokenizer and embedder to further pass through the RNN and CNN layers

5.1.2. Transformer Model Based Structure

RoBERTa[9], Refer Figure 2, model takes care of capturing contexts in long sequences of texts and co-dependency of tokens frequently coming together in a sentence. It uses RoBERTa Tokeniser and RoBERTa Embedding to encode the entire sentence and then transfer learning is performed using our custom data. The purpose of using RoBERTa is primarily to introduce a multi head attention mechanism which helps in capturing the contextual information of longer sequences efficiently, compensating for the drawbacks presented while using BiLSTM+CNN [8] network. The Named entity task in our experiment is to classify tokens into SIO annotations of either custom tags for domain specific tasks, or IOB annotated POS tags for benchmarking purposes. These tags are then used for segmentation using the Inside and Outside tags and give a final output of chunked phrases of a sentence.

The accuracy of the chunks is hence directly proportional to the accuracy of the Token classification task. Attention mechanism helps classify the chunk position correctly, it failed to classify simpler tags, and misclassified tokens with similar Part of speech tags in lengthier chunks.

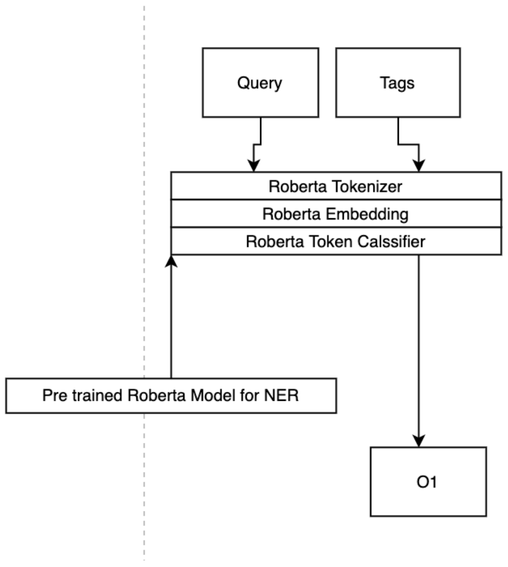


Figure 6. Transfor Model: RoBERTa: This model produces Output 1 or O1

5.1.3. Ensemble model

Using an Ensemble learning Model will help compensate for both the models' flaws, and hence it gives better performance. Multi head attention mechanisms helped us to resolve infinite context by taking attention forever pairwise tokens,, BiLSTM is very good at preserving local context and CNN is quite useful in local optimisation i.e recognition of the original chunk in a neighbourhood. A weighted average of classification scores of RoBERTa and BiLSTM + CNN are then used for final Multi class classification.

We trained the Ensemble Model for 150 epochs using the Adam Optimiser and a learning rate of 0.01. The model converged to an F1 score of 97.3 which is higher than either of the models described individually. We can thus conclude with this experiment that ensemble training using custom tokenizer encapsulates more information required for a Named Entity Recognition Task.

$$\text{final classification scores} = w1*O1 + w2*O2 \qquad \text{.... Eqn. 2}$$

where,

O1, O2 are outputs from Model A and Model B respectively.
w1, w2 are the weights associated with it.

5.2. Application of Ensemble Deep neural framework in building a domain based chunker with custom SIO tags (TASK-2)

Extending the use of this Ensemble of Deep Neural Networks in our domain specific task we replace the IOB annotated POS chunk tags with our custom context phrase SIO tags. The Input is hence the sentences generated using the domain specific data catalog, including a diverse range of expectant queries with shorter phrase type queries (example: young men) , Attribute collection type queries (example: facebook, instagram gamer) and Longer well structured sentence type query (example: Men aged 35 living in Texas using Facebook and Netflix), and hence try to mimic the range of queries that are fed as input from client briefings which varies from structured to highly unstructured sentence format.

A large number of production rules are used to bring in diversity to the collection of queries, which are formulated after drawing inferences from real world search queries. Production rules can be later added, subtracted or modified as per the feedback received from user experiences. The sentences generated are segregated into training and evaluation sentences groups for the token classification task and the evaluation metrics is the token level classification F1 score calculated for correctly classifying correct chunk tags, which is a harmonic mean of classification precision and recall. The F1 score for scaled data and tuned hyperparameters are mentioned in Table 3.

6. OBSERVATIONS, RESULTS, BENCHMARKS

For the Task-1, the training sentences are preprocessed into a [token]-[tag]-[sentence-id] format for training. Every query is whitespace tokenized, every token hence obtained is mapped to its annotated tag and a sentence id corresponding to the query. A GPU Nvidia Tesla K80 is used for training. Naturally, for this kind of task the evaluation metric classification F1 score is taken into account because the data set has imbalanced tags and accuracy might therefore be biased towards the majority tag.

Comparing our results with the present state of the art for chunking[10], on conll 2000 data set Refer Table 1, the model exceeds the state of the art neural net model which uses Bi-LSTM in conjunction with CNN for encoder-decoder labeling and pointer framework for segmentation as mentioned before. Our model obtains an F1 score of 97.3 exceeding the preceding State of the Art 94.72 for conll 2000 dataset. The comparison is also done on 23 labels for 9000 training sentences and 900 testing sentences.

Table 1. Comparative Analysis of Algorithms. Dataset(ConLL 2000 english).

Model	**ConLL 2000**
RoBERTa +(Bi-LSTM+CNN)	97.3
RoBERTa	96.76
BERT	95.64
Bi-LSTM+CNN	94.72

To Evaluate and validate our experimentation described in Task-2, we segregate the Templates collection itself into training and testing sets, hence automatically segregating the sentences generated using these two sets of templates. We use these out of template generated queries to ensure the validity of evaluation metrics. The annotated sentences are preprocessed similar to task 1 where a map of [token]-[sentence-id]-[tag] is created after whitespace tokenization. The [token] and [sentence-id] here are vectorized and fed in as input to the neural net model and [tag] is fed as expected output label. We followed a 85% to 15% segregation of training to test query data. The model is trained for 6 epochs with a batch size of 64 and uses the same computing resource as Task-1, i.e. An Nvidia Tesla K80 GPU,

The model gives a test classification F1 score of 96.97% on our out of training template custom generated data. The test data is preprocessed in the same way as training data and the model predicts labels for these token vectors. These labels are compared to the ground truth labels generated using test templates and precision, recall and F1 score is calculated using the confusion matrix hence obtained.

After training the model, a layer of segmentation is deployed to chunk the phrases and contexts together to further pass onto the last layer of the model to map the relevant context and phrases together. The result of the model hence is a dictionary where key, value pairs are the context phrase(s) pairs.

Table 3 compares how changing the number of production rules, size of data and no of epochs affects the model performance. With a jump from 8 to 19 production rules we added the tag "O" and from 20 to 22 production rules, we had introduced the tag "DIFF" and hence the increase in no of queries per template. The consistency in F1 score owes to the increase in data size in sync with the increase in diverse rules. The number of epochs in the most recent training is less than its previous training to avoid over fitting.

Table 3: Effect of Tuning the Hyper parameters on F1 score

No of PR	Train sentences	Test sentences	Accuracy	F1 score	Batch Size	Epochs
8	60000	2000	88.49	90	8	2
19	60000	15000	87.2	95.9	32	5
20	64000	12000	88.57	95.9	64	9
22	144000	24000	87.3	96.97	64	6

7. CONCLUSIONS

In this paper, we tried to tackle two specific problems. In the first part, we aim to highlight the significance of the ensemble in building the best chunking model. In the ensemble, one part was the state-of-the-art transformer-based model. It uses an attention-based framework to understand the semantics of long-length sentences better. We found that the transformer-based model does the misclassifications at various Symbols, and Punctuations, and Nouns and Proper nouns. To compensate for this in the second part, we provided POS tags of the token explicitly to the BILSTM+CNN architecture. The CNN part optimizes the local context well and The BiLSTM with pos tag tries to capture the context dependency with the surrounding word better. We experimented with this ensemble model on various open datasets for chunking tasks and found that this ensemble architecture broke the previous state-of-the-art. In future work, we would apply this architecture to different GLUE tasks to prove the robustness of the model.

In the second part, we solved the tedious task of building a domain-based chunker from scratch without any annotated data previously available. We proved that the grammar-based annotation was able to explain the diversity and complexity of the domain with few production rules and the underlying domain data catalog. We achieved a significant result of 96.67% F1 score accuracy in classifying the tokens into the correct chunk tag class on the sentences which were generated out of a template of unknown grammar rules. Previously this kind of problem was solved by employing a huge number of resources with domain knowledge to do manual annotation. With the innovation explained in our paper, getting domain-based annotated data and then building a customized chunker becomes easily accessible for everybody. In the future, we would experiment to replicate the same for different domains to prove the robustness of the model.

REFERENCES

[1] Nothman And James R. Curran And Tara Murphy, “Transforming Wikipedia Into Named Entity Training Data,” 2008.

[2] V. Yadav and S. Bethard, “A survey on recent advances in named Entity Recognition from deep learning models,” arXiv [cs.CL], 2019.

[3] E. Muszyńska, “Graph- and surface-level sentence chunking,” in Proceedings of the ACL 2016 Student Research Workshop, 2016.

[4] A. Siddharthan, "Complex Lexico-syntactic Reformulation of Sentences Using Typed Dependency Representations", ACL Anthology, 2021. [Online]. Available: https://www.aclweb.org/anthology/W10-4213.

[5] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, “Neural Architectures for Named Entity Recognition,” arXiv [cs.CL], 2016

[6] D. Nadeau and S. Sekine, “A survey of named entity recognition and classification,” in Benjamins Current Topics, Amsterdam: John Benjamins Publishing Company, 2009, pp. 3–28.

[7] T. Kudo and Y. Matsumoto, “Chunking with Support Vector Machines,” J. Nat. Lang. Process., vol. 9, no. 5, pp. 3–21, 2002.

[8] F. Zhai, S. Potdar, B. Xiang, and B. Zhou, “Neural Models for Sequence Chunking,” arXiv [cs.CL], 2017.

[9] Y. Liu et al., “RoBERTa: A robustly optimized BERT pretraining approach,” arXiv [cs.CL], 2019.

[10] R. Chalapathy, E. Zare Borzeshi, and M. Piccardi, “An investigation of recurrent neural architectures for drug name recognition,” in Proceedings of the Seventh International Workshop on Health Text Mining and Information Analysis, 2016.

[11] R. Collobert and J. Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning,” in Proceedings of the 25th international conference on Machine learning - ICML ’08, 2008.

[12] F. Dernoncourt, J. Y. Lee, and P. Szolovits, “NeuroNER: an easy-to-use program for named-entity recognition based on neural networks,” in Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2017..

[13] R. Panchendrarajan and A. Amaresan, "Bidirectional LSTM-CRF for Named Entity Recognition", ACL Anthology, 2021. [Online]. Available: https://www.aclweb.org/anthology/Y18-1061.

[14] Y. Li, K. Bontcheva, and H. Cunningham, “SVM based learning system for information extraction,” in Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 319–339.

[15] W. Etaiwi, A. Awajan, and D. Suleiman, “Statistical Arabic name entity recognition approaches: A survey,” Procedia Comput. Sci., vol. 113, pp. 57–64, 2017.

[16] E. J. Otoo, D. Rotem, and S. Seshadri, “Optimal chunking of large multidimensional arrays for data warehousing,” in Proceedings of the ACM tenth international workshop on Data warehousing and OLAP - DOLAP ’07, 2007.

[17] H. Sharma, "Survey of Research on Chunking Techniques", Semanticscholar.org, 2021. [Online]. Available: https://www.semanticscholar.org/paper/Survey-of-Research-on-Chunking-Techniques-Sharma/ced929651b222d3b489df1c25ed9c801c7c3e749.

[18] P. S. Rosenbloom and J. Aasman, “Knowledge level and inductive uses of chunking (EBL),” in Soar: A Cognitive Architecture in Perspective, Dordrecht: Springer Netherlands, 1992, pp. 219–234.

[19] A.Vaswani et al., “Attention is all you need,” arXiv [cs.CL], 2017.

[20] M. Marcus, B. Santorini and M. Marcinkiewicz, "Building a Large Annotated Corpus of English: The Penn Treebank", ACL Anthology, 2021. [Online]. Available: https://www.aclweb.org/anthology/J93-2004.

[21] E. F. Tjong Kim Sang and S. Buchholz, “Introduction to the CoNLL-2000 shared task: Chunking,” in Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning -, 2000.

[22] Y. Zhu et al., “Aligning books and movies: Towards story-like visual explanations by watching movies and reading books,” in 2015 IEEE International Conference on Computer Vision (ICCV), 2015.

[23] Wikipedia: Database download

[24] Mohapatra, Nilamadhaba, Namrata Sarraf, and Swapna sarit Sahu. "Ensemble Model for Chunking." CS & IT Conference Proceedings. Vol. 11. No. 8. CS & IT Conference Proceedings, 2021.

[25] Bateman, John. "Sentence generation and systemic grammar: an introduction." Iwanami Lecture Series:Language Sciences, Iwanami Shoten Publishers, Tokyo (1997).

[26] Savitch, Walter J. "Context-sensitive grammar and natural language syntax."The Formal complexity of natural language. Springer, Dordrecht, 1987. 358-368.

[27] Asveld, Peter RJ. "Generating all permutations by context-free grammars in Chomsky normal form. "Theoretical Computer Science 354.1 (2006): 118-130.

[28] Engelfriet J. A Greibach normal form for context-free graph grammars. In International Colloquium on Automata, Languages, and Programming 1992 Jul 13 (pp. 138-149). Springer, Berlin, Heidelberg.

[29] De Smedt, Koenraad, and Gerard Kempen. "Segment grammar: A formalism for incremental sentence generation." Natural language generation in artificial intelligence and computational linguistics. Springer, Boston, MA, 1991. 329-349.

[30] Magassouba, Aly, Komei Sugiura, and Hisashi Kawai. "Multimodal attention branch network for perspective-free sentence generation." Conference on Robot Learning. PMLR, 2020.

[31] Liu, Jiangming, Shay B. Cohen, and Mirella Lapata. "Text Generation from Discourse Representation Structures." Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2021.

[32] Zhu, Yukun, et al. "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books." Proceedings of the IEEE international conference on computer vision. 2015.

[33] Wen, Tsung-Hsien, et al. "Semantically conditioned lstm-based natural language generation for spoken dialogue systems." arXiv preprint arXiv:1508.01745 (2015).

[34] Oh, Alice, and Alexander Rudnicky. "Stochastic language generation for spoken dialogue systems." ANLP-NAACL 2000 Workshop: Conversational Systems. 2000.

[35] Zhang, Haoyu, Jianjun Xu, and Ji Wang. "Pre Training-based natural language generation for text summarization." arXiv preprint arXiv:1902.09243 (2019).