Information Extraction on Tourism Domain using SpaCy and BERT

Chantana Chantrapornchai[1] and **Aphisit Tunsakul**[2]

ABSTRACT: Information extraction is a basic task required in document searching. Traditional approaches involve lexical and syntax analysis to extract words and part of speech from sentences in order to establish the semantics. In this paper, we present two machine learning-based methodologies used to extract particular information from full texts. The methodologies are based on the tasks: name entity recognition (NER), and text classification. The first step is the building training data and data cleansing. We consider a tourism domain using information about restaurants, hotels, shopping, and tourism. Our data set was generated by crawling the websites. First, the tourism data is gathered and the vocabularies are built. Several minor steps include sentence extraction, relation and name entity extraction for tagging purposes. These steps are needed for creating proper training data. Then, the recognition model of a given entity type can be built. From the experiments, given review texts on the tourism domain, we demonstrate how to build the model to extract the desired entity, i.e., name, location, or facility as well as relation type, classifing the reviews or the use of the classification to summarize the reviews. Two tools, SpaCy and BERT, are used to compare the performance of these tasks. The accuracy on the tested data set on the name entity recognition for SpaCy is upto 95% and for BERT, it is upto 99%. For text classification, BERT and SpaCy yield accuracies of around 95%-98%.

Keywords: Name Entity Recognition, Text Classification, BERT, SpaCy

DOI: 10.37936/ecti-cit.2021151.228621
Received December 11, 2019; revised March 18, 2020; accepted June 18, 2020; available online January 5, 2020

1. INTRODUCTION

Typical information search in the web requires text or string matching. When the user searches for the information, the search engine returns the relevant documents that contain the matched string. The users need to browse through the associated links to find whether the website is in the scope of interest, which is very time consuming.

To facilitate the user search, a tool for information extraction can help the user find relevant documents. A typical approach requires processes such as lexical analysis, syntax analysis, semantic analysis. Also, from a set of keywords in the domains, word frequency and word co-occurences are calculated. Different algorithms propose different rules for determining co-occurences. These approaches are mostly hand-coded rules which may highly rely on languages and domains used to infer the meaning of the documents.

Using a machine learning approach can facilitate the rule extraction process. The pre-trained language model embeds the existing representations of words and can be used to extract their relations automatically. Nowadays, there are many pre-trained language models such as BERT [1]. It provides a contextual model, and can also be fine-tuned for a specific language and/or domain. Thus, it has been used popularly as a basis and extended to perform many language processing tasks [2].

In this paper, we focus on the machine learning approach to perform the basic natural language processing (NLP) tasks. The tasks in question are name entity extraction and text classification. The contribution of our work is as follows:

- We construct an approach to perform information extraction for these two tasks.

[1]Faculty of Engineering, Kasetsart University, Bangkok, Thailand, E-mail: fengcnc@ku.ac.th
[2]National Science and Technology Development Agency(NSD),Pathum Thani, Thailand.

This work was supported in part by Kasetsart University Research and Institute funding and NVIDIA hardware grant.

• We demonstrate the use of two frameworks: BERT and SpaCy. Both provide pre-trained models and provide basic libraries which can be adapted to serve the goals.
• A tourism domain is considered since our final goal is to construct the tourism ontology. Tourism data set is used to demonstrate the performance of both methods. The sample public tagged data is provided as a guideline for future adaption.

It is known that extracting data into the ontology usually requires lots of human work. Several previous works have attempted to propose methods for building ontology based on data extraction [3]. Most of the work relies on the web structure of documents [4-6]. The ontology is extracted based on HTML web structure, and the corpus is based on WordNet. The methodology can be used to extract required types of information from full texts for individuals insertion to the ontology or for other purposes such as tagging or annotation.

When creating machine learning models, the difficult part is the data annotation for training data set. For name entity recognition, the tedious task is to do corpus cleansing and do the tagging. Text classification is easier because topics or types can be used as class labels. We will describe the overall methodology which starts from data set gathering by the web information scraping process. The data is selected based on the HTML tag for corpus building. The data is then used for model creation for automatic information extraction tasks. The preprocessing for training data annotation is discussed. Various tasks and tools are needed for different processes. Then, the model building is the next step where we demonstrate the performance of two tools used in information extraction tasks, SpaCy and BERT.

Section 2 presents the backgrounds and previous works. In Section 3, we describe the whole methodology starting from data preparation for information tagging until model building. Section 4 describes the experimental results and discussion. Section 5 concludes the work and discusses the future work.

2. BACKGROUNDS

We give examples of the existing tourism ontology to explore the example types of tourism information. Since we focus on the use of machine learning to extract relations from documents, we describe machine learning and deep learning in natural language processing in the literature review.

2.1 Tourism ontology

Many tourism ontologies have been proposed previously. For example, Mouhim et al. utilized the knowledge management approach for constructing an ontology [7]. They created a Morocco tourism ontology. The approach considered Mondeca tourism ontology in OnTour [8] was proposed by Siorpaes et al. They built the vocabulary from a thesaurus obtained from the United Nation World Tourism Organisation (UNWTO). The classes as well as social platform were defined. In [9], the approach for building e-tourism ontology is based on NLP and corpus processing which uses POS tagger and syntactic parser. The named entity recognition method is Gazetter and Transducer. Then the ontology is populated, and consistency checking is done using OWL2 reasoner.

STI Innsbruck [10] presented the accommodation ontology. The ontology was expanded from GoodRelation vocabulary [11]. The ontology contains the concepts about hotel rooms, hotels, camping sites, types of accommodations and their features, and compound prices. The compound price shows the price breakdowns. For example, the prices show the weekly cleaning fees or extra charges for electricity in vacation homes based on metered usage. Chaves et al. proposed Hontology which is a multilingual accommodation ontology [12]. They divided it into 14 concepts including facility, room type, transportation, location, room price, etc. These concepts are similar to QALL-ME [13], as shown in Table 1. Among all of these, the typical properties are things such as name, location, type of accommodation, facility, price, etc. Location, facility, and restaurant are examples used in our paper for information extraction.

Table 1: *Concept mapping between Hontology and QALL-ME [13].*

Hontology	QALL-ME
Accommodation	Accommodation
Airport	Airport
BusStation	BusStation
Country	Country
Facility	Facility
Location	Location
MetroStation	MetroStation
Price	Price
Restaurant	Restaurant
RestaurantPrice	**GastroPrice**
RoomPrice	RoomPrice
Stadium	Stadium
Theatre	Theatre
TrainStation	TrainStation

2.2 Machine Learning in NLP

There are several NLP tasks including POS (part-of-speech), NER (named entity recognition), SBD (sentence boundary disambiguation), word sense disambiguation, word segmentation, entity relationship identification, text summarization, text classification, etc. Traditional methods for these tasks requires the lexical analysis, syntax analysis and a rule-base for

hand-coded relations between words. The rule-base must match the words within the varying window sizes, which varies a case by case basis. In the machine learning approach, there is no need to hand-code the rules. The requirement for lexical analysis and syntax analysis is very small. The tagging (annotation) of words is needed and some data cleansing is required. The tagging also implies the type of entity (word/phrase). Tagging relationships may also be possible. The model will learn from these tags. The accuracy of the model depends on the correctness and completeness of the training data set, while the rule-bases approach's accuracy depends on the correctness and completeness of the rules.

The rule-based approach is very fragile and sensitive to a language structure. Recently, we found several works applying machine learning to NLP tasks [14]. Typical machine learning is used to learn language features and build a model to solve these tasks. Common models are Naive Bayes, SVM, Random Forest, Decision Tree, Markov model, statistical model, and deep learning model.

2.3 SpaCy

SpaCy [15] provides an open-source library and neural network model to perform basic NLP tasks. The tasks include processing linguistic features such as tokenization, part-of-speech tagging, rule-based matching, lemmatization, dependency parsing, sentence boundary detection, named entity recognition, similarity detection, etc. The model is stochastic and the training process is done based on gradient and loss function. The pretrained model can be used in transfer learning as well.

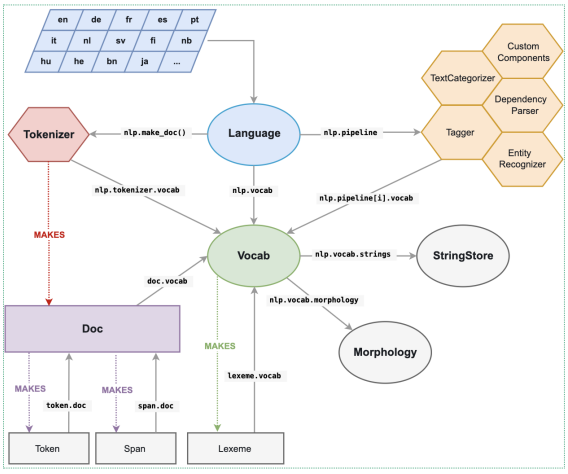


Fig.1: *SpaCy architecture [16]*

Figure 1 shows the library architecture. The keys are Doc and Vocab. Doc contains a sequence of tokens. *nlp* is the highest level which involves many objects including Vocab and Doc. It has a pipeline to many tasks such as tagger, dependency parser, entity recognition, and text categirzation.

The use of the Python library begins with *import* and *nlp* is used to extract linguistic features. The first example is to extract part-of-speech. Examples are shown in SpaCy web pages [17].

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Apple is looking at buying
U.K. startup for $1 billion")

for token in doc:
    print(token.text, token.lemma_, token.pos_,
          token.tag_, token.dep_,
           token.shape_,
           token.is_alpha, token.is_stop)
```

The linguistic features are in the fields in the *print* statements: *token.lemma_, token.pos_, token.tag_, token.dep_, token.shape_, token.is_alpha, token.is_stop*

The part of speech is in *token.pos_*. The tag is *token.tag_* and the token dependency type is in *token.dep_*. In the example, apple is noun subject, and it is a proper noun.

```
Apple Apple PROPN NNP nsubj Xxxxx True False
is be AUX VBZ aux xx True True
looking look VERB VBG ROOT xxxx True False
at at ADP IN prep xx True True
buying buy VERB VBG pcomp xxxx True False
U.K. U.K. PROPN NNP compound X.X. False False
startup startup NOUN NN dobj xxxx True False
for for ADP IN prep xxx True True
$ $ SYM $ quantmod $ False False
1 1 NUM CD compound d False False
billion billion NUM CD pobj xxxx True False
```

The next example extracts the name entity.

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Apple is looking at buying U.K.
startup for $1 billion")
for ent in doc.ents:
    print(ent.text, ent.start_char,
    ent.end_char, ent.label_)
```

The results are shown in the fields *ent.start_char* for the starting position, *ent.end_char* for the ending position, and *ent.label_* for the type of entity as following. Apple is an ORG and at positions 0-5.

```
Apple 0 5 ORG
U.K. 27 31 GPE
$1 billion 44 54 MONEY
```

The current architecture of the SpaCy named entity recognition model is not published yet. To our knowledge, it is a statistical model utilizing multilayer perceptron. A stack of weighted bloom embedding layers merged with neighboring features together are used. The transition-based approach is used for training (`https://www.youtube.com/watch?v=sqDHBH9IjRU`).

2.4 Deep Learning and BERT

In deep learning, the use of a deep network is for the purpose of learning features. The common model for this task is Recurrent Neural Network (RNN) [18]. The RNN captures the previous contexts as states and is used to predict the outputs (such as next predicted words). RNN can be structured many ways, such as a stack, a grid, as well as supporting bidirectional access to learn from left and right contexts. The RNN cell can be implemented by LSTM (Long Short Termed Memory) or GRU (Gated Recurrent Unit) to support the choice of forgetting or remembering.

One typical model is the Seq2Seq model which applies bidirectional LSTM cells with the attention scheme [19]. The example application is abstractive text summarization [20]. The extractive text summarization can be based on the text ranking approach (similar to page ranking) to select top rank sentences [21]. The text summarization is an important task for many NLP applications but the current approaches still yield the results with low accuracy for real data sets. This task requires a large and specific training data set to increase accuracy.

Traditional RNNs have a major drawback since it may not be suitable to apply the pre-trained weights. This is because the pre-trained model usually is obtained from different domain contexts. Retraining it will make it soon forget the previous knowledge and adapt itself to a new context. Recently, Google proposed a pre-trained transformer, BERT, which is bidirectional and can be applied to NLP tasks [1]. The concept of the transformer does not rely on shifting the input to the left and to the right like in Seq2Seq. The model construction also considers sentence boundaries and relationships between sentences. The model derives the embedding representation of the training data.

BERT has been used to perform many tasks such as text summarization [22] where both extractive and abstractive approaches are explored. The extension is called BERTSUMEXt where CLS is inserted multiple places to learn interval embedding between sentences. The data set is news which comes from NYT,CNN/Daily Mail. In [23], BERT was used for extractive summarization for health informatic lectures as a service. BERT embedding is used, and then K-Means clustering identifies the closet sentences to the summarry. Munikar et.al. applied BERT for fine-grained sentiment text classification [24]. The authors used Stanford Sentiment Treebank (SST) data set where there are five labels: 0 (very negative), 1 (negative), 2 (neutral), 3 (positive), and 4 (very positive). They applied dropout and softmax layers on top of BERT layers. DocBERT is another approach for document classification [25]. They adopted BERT_{base} and BERT_{large} for fine tuning and then applied knowledge distrillation to transfer the knowledge to smaller BiLSTM with fewer model parameters. They experimented on data sets Reuters-21578, AAPD, IMDB, and Yelp 2014.

NER is a basic task that is useful for many applications relevant to information searching. In order to find the documents related to the user search, the semantic of the documents must be analyzed. The meaning of phrases or words as well as their parts of speech must be known. Name entity recogniton is the task which can help achieve this goal. To discover the named entity of a given type, lots of training data is used. The training data must be annotated with proper tags. POS tagger is the first required one, since part-of-speech is useful for discovering types of entities. Other standards of tagging are IO, BIO, BMEWO, and BMEWO+ [26]. These are used to tag positions of tokens inside word chunks. The tagging process is a tedious task, so an automatic process is needed. Machine learning can, therefore, be applied to the named entity recognition to help automatic tagging [27]. Recent works in using BERT in the NER task [28-30] which consider BERT-NER in an open domain are HAREM I Portugese language, and Chinese Medical Text respectively. In our work, we focus on a tourism data set. For NER task, the tags for the named entity types we look for have to be defined and we train the model to recognize them. Also, we can train the parser to look for the relation between our entity types. Then, we can extract the desired relation along with the named entity.

3. METHODOLOGY

We started by collecting a tourism data set in Thailand by area and performing annotation. Since we focus on finding the concept of location, organization, and facility in the tourism data set, the approaches that we can apply are the following:

- Use the classification approach. We classify the sentences by the concept type, for example, location (0), organization (1), facility (2). The classification may be obtained by examining the HTML tag when we do web scraping. The classification is also used as a tool for the summarization approach. The summary label is defined as a type of heading. It is also the concept in this case. HTML tags can be used for annotation.
- Use the NER approach. This requires more work on annotation. It also depends on the type of model used for NER Task. In this approach, we are able to find the boundaries of words that contain the concept. This is the most complicated one for data annotation among them. We will discuss this more later.

Figure 2 presents the overall methodology of preparing data for creating the models for the NER tasks. The difficult part is to label the data automatically, which will also be useful for other similar tasks. In Section 4, we demonstrate the accuracy results of the models after training.

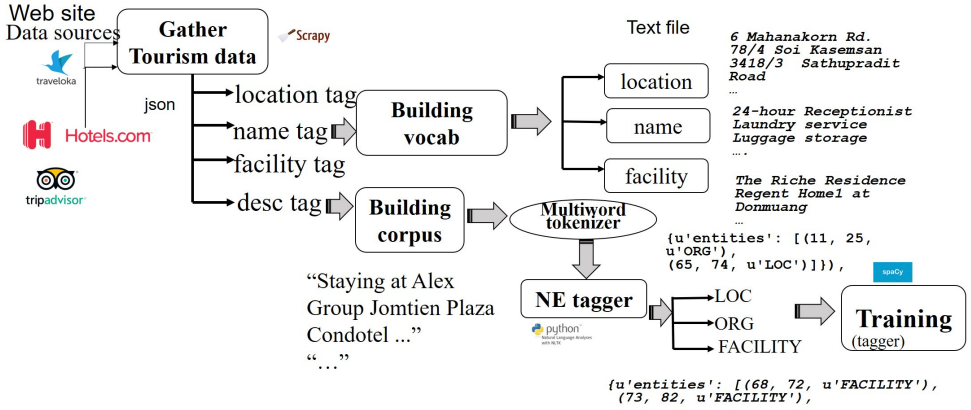


Fig.2: *Overall process for creating training model.*

3.1 Data gathering

To prepare the tourism data, we crawled data using the Scrapy library (`https://scrapy.org/`) in Python. There are two data sets here. The first data set is from the crawling of three websites: Tripadvisor, for a total of 10,202 hotels, Traveloka, for a total of 5,430 hotels, and Hotels.com, for 11,155 hotels. For each website, eight provinces are considered, including Bangkok, Phuket, Chaingmai, Phang-nga, Chonburi, Suratani, Krabi, and Prachuap Khiri Khan. For each province, we collected six features: name, description, address, facility, nearby, and review.

An example of the data obtained in JSON format is:

```
[{ "address": "1/1 Moo 5 Baan Koom, Doi
 Angkhang, Tambon Mae Ngon, Amphur Fang ,
  Mon Pin, Fang  District, Chiang Mai,
  Thailand, 50320",
 "description": ",Staying at Angkhang Nature
 Resort is a good choice when you are
 visiting Mon Pin.This hotel is  ....
 "facility": "WiFi in public area,
 Coffee shop,
 Restaurant,Breakfast,...
 "name": "B.M.P. By Freedom Sky",
 "nearby": "Maetaeng Elephant Park,
 Maetamann Elephant Camp,Mae Ngad Dam
 and Reservoir,Moncham",
"review": "" }
{ ...}]
```

In the second data set, we preprared the data by going through Tripadvisor website. We are interested in the review part. We aim to use this data set as a demonstration of text classification as well. The review can also be used in NER tasks. We extracted an information about 5,685 restaurants, 10,202 hotels, and 3,681 tours. For each data set, eight province are considered, including Bangkok, Phuket, Chaingmai, Phang-nga, Chonburi, Suratani, Krabi, and Prachuap Khiri Khan. For each record, we collected fourteen features: name, description, address, star-rating, key-tag, votes, review1, review2, review3, review4, tag-re1, tag-re2, tag-re3, and tag-re4. The review data obtained is in JSON:

```
[{"star_rating": 4.5,
  "description": "PRICE RANGE,THB 216 -
  THB 556,CUISINES ...",
  "address": "109 SuriWong Road,
  Patpong/Silom Bang Rak, Patpong,
  Bangkok 10500 Thailand",
  "votes": "849 reviews",
  "review3": "This place is NOT to
  be missed. 3 words...Wing.
  He took the time to chat with us and
  we appreciate a good
  conversation! We had
  other dishes as well that were...",
  "review2": "The food, service and
  ambience is just perfect.
  A place of real personality,
  quality and charm...",
  "review1": "Lovely service,
  an excellent vibe or
  great food, value for money
  and super friendly staff ...",
  "key_tag": "All reviews ,thai food ,
  orange curry ,wing bean salad ,
  massaman curry ...",
  "review4": "One of the best Restaurant
  Lounge Bar in Bangkok center, offering
  really good value for money with
  excellent service together
  with food quality, one...",
  "tag_re4": "OUTSTANDING",
  "tag_re3": "Delicious and Close to Fun!",
  "tag_re2": "Go here first!",
  "tag_re1": "Excellent meal!",
  "name": "Oasis Lounge"}
{ ...}]
```

3.2 Vocabulary building

The key task of creating the training data is to generate tags for the tourism data. The building of the vocabularies, which are sets of special keywords in our domain, is required and then saved in text files. These keywords can be multiple words or chunks. They are useful for tokenizing word chunks (by multiword tokenizer) for a given sentence. For example, "Thong Lor, JW Marriott, Room service" should be recognized as a single chunk. To build our multiword vocabulary files we extracted the values from these JSON fields: 'name', 'location', 'nearby', and 'facility'.

Table 2: *Corpus statistics.*

location	facility	nearby	hotel name	all	all(nodup)
17,660	17918	18,822	41,168	58,828	37,107

The values in these fields, separated by commas, are split and saved into each text file: 1) location, and nearby are saved as `location-list.txt` 2) name is saved as `hotel-name-list.txt` 3) facility is saved as `facility-list.txt`, as shown in Figure 2 in "Building vocab". The number of values for keywords and total vocabularies are displayed in Table 2. Column 'all (nodup)' shows the all combined vocabularies after removing duplicates. Then, a multi-word expression tokenizer is built using Python library (`nltk.tokenize.mwe`). We can use other multi-word approachs to help extracting noun phase representing location name, organization name, etc.

3.3 Handling multiword vocabularies

SpaCy also provides the nounchunk library to split sentences into noun phrases. The noun phrases can be selected and added to a keyword textfile. They can be used together with the multiword tokenizer. The additional step is depicted in Figure 3.

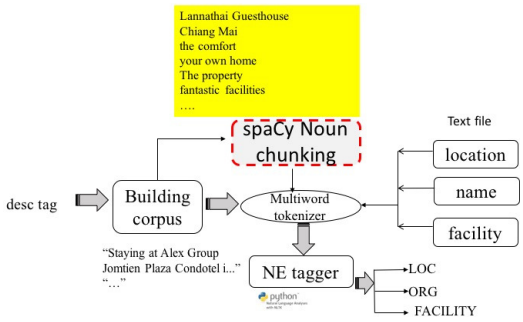


Fig.3: *Adding pipeline of noun phase chunking.*

3.4 Training data annotation

The model is built for recognizing these new entities and keywords. It is trained with location name, hotel name, and facility tags. For creating the training data for SpaCy (`http://spacy.io`), the training data must be converted to a compatible form as its input. Depending on the goal of training model, the label input data is transformed properly. The interesting named entity types are LOC, ORG, and FACILITY. LOC identifies the location or place names. ORG refers to hotel or accommodation names. FACILITY refers to facility types.

We added our vocabulary keywords since our location name and hotel name (considered as ORG) are noun-phases and specific to our country. For FACILITY, we need to create our new named entity label for the name of the facility in the hotel.

3.4.1 Extracting training sentences

Our raw data files contain many sentences for each section. We cannot use whole paragraphs and annotate them since each paragraph is too long and there are lots of irrelevant sentences/words which can create a lot of noise in training data. Manually extracting sentences from description fields is a time-consuming task. We have to specifically find the sentences that contain these named entities. The sentences should be selected from the paragraphs which contain either hotel names, locations, or facilities.

When crawling the data source, we intend to obtain a list of paragraphs describing a particular hotel or hotel review. This information is useful for creating corpus and review summary.

Text summarization is a common technique used in on basic NLP tasks as well. The extracted summary can be obtained either from abstractive or extractive methods. For our case, we extract sentences for creating training raw corpus for named entity training. An extractive summary is useful for pulling important sentences and using these sentences for tagging. The pipeline for raw corpus building can be modified as shown in Figure 4 where TextRank can be applied to pull the important sentences.

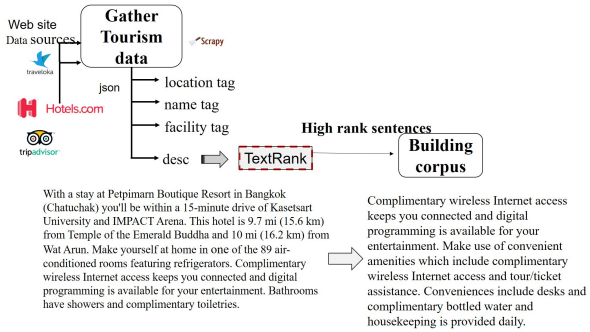


Fig.4: *Adding pipeline of TextRank.*

3.4.2 Relation type extraction

When we need to discover the relationship between named entities in a sentence, we may use the model which is trained to recognize the relationships between two words or two chunks. There are two approaches for recognizing the relations suggested by SpaCy model. The first approach is to use a dependency parser in SpaCy. The second approach is to train our model to recognize the relationship keywords as new tags.

In the first approach, the two words are defined to be related as as dependency (deps) as in the following example.

```
("Conveniences include desks and complimentary
bottled water",
{'heads': [0, 0, 0, 0, 6, 6,0],
# index of token head and
'deps': ['ROOT', '-', 'FACILITY',
'-', 'TYPE', 'TYPE','FACILITY']
```

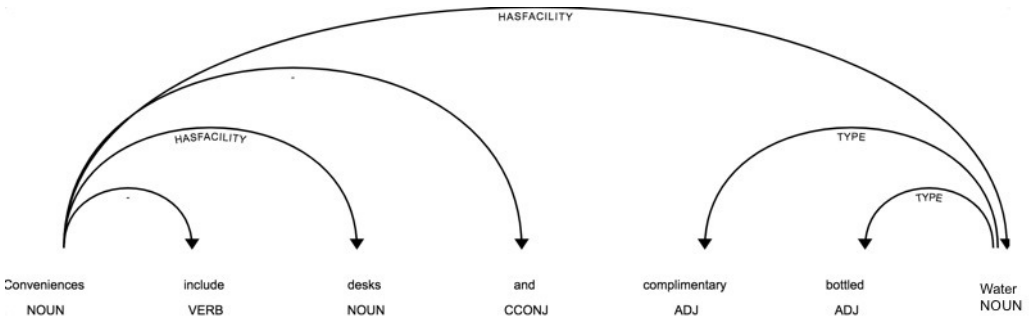


Fig.5: *Relation marking*

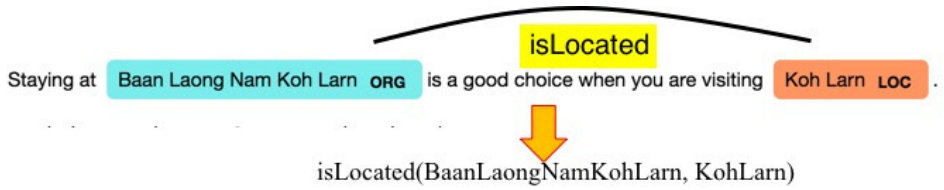


Fig.6: *Relation capturing part I.*

```
# dependency type between pair
 }),
....
```

In this above example, 'heads' is a list whose length is equal to the number of words. 'deps' is the list of name relations. Each element refers to its parents. For example, 0 at the last element (water) refers to the relation 'FACILITY' to 'convenience' and 6 refers to the modifier 'TYPE' of 'water'. '-' means no relation (or ignored). Obviously, it requires effort for manual annotations. Figure 5 shows dependency graph from SpaCy for this example. The dependency between "conveniences" and "desk", as well as "water" and "HAS FACILITY". Figure 6 shows the example of relation in our raw corpus for the property `isLocated` which is used in tourism ontology concept.

In Figure 7, there can be lots of relations in one sentence, making it harder to annotate. The sentence exhibits both `isLocated` and `hasFacility` properties. Figure 8 is the new pipeline following the NE recognition stage to train our relationship model. Only sentences with NEs are extracted, and the annotation of a relation dependency for each sentence can be processed.

3.4.3 Integrating all tags

To recognize both NEs and relation names at the same time, we added relation tags to the training data. We created indication words which imply the location, organization, facility, and relation name in the sentences. The sentences were split into selected phrases based on the named entities, according to the corpus, tagged as LOC, ORG, LOCATEAT, or FACILITY. We recorded the starting index position, ending index position) for each word chunk found.

In the example, "Staying at @ Home Executive Apartment is a good choice when you are visiting Central Pattaya.", the boundary of words for tag LOCATEAT is from position 0-10, which is "Stay at". Position 11-37, "@ Home Executive Apartment" is ORG. Position 77-92, "Central Pattaya" is LOC, etc.

The annotation is formated in JSON as:

```
'text': Staying at @ Home Executive
 Apartment is a good choice when
  you are visiting
   Central Pattaya.
'entities': [(11,37,ORG), (77,92,LOC)]
```

In the above example, there are two entities, '@ Home Executive ' which begins at character index 11 and ends at index 37, and its type is ORG name (hotel name). 'Central Pattaya' begins at character index 77 and ends at index 92, and its type is LOC.

3.4.4 BIO tagging

For BERT-NER task, BIO tagging is required. BIO tagging is commonly used in NLP tasks [31], though other tagging approaches may be used. The training data is adjusted as seen in Figure 9. We transformed the input raw corpus from SpaCy to its form. The sentence is split into words and each word is marked with POS tagger. Column 'Tag' is our tag name for each word. We have to split the multiword named entity and tag each word since we have to use BERT tokenizer to convert to word ID before training. In the figure, our hotel name starts at '@', so we use the tag B-ORG (beginning of ORG) and 'Home' is the intermediate word after it (I-ORG). BERT needs to convert the input sentences into lists of word IDs along with the list of label IDs. The lists are padded to equal size before sending them to train the model.

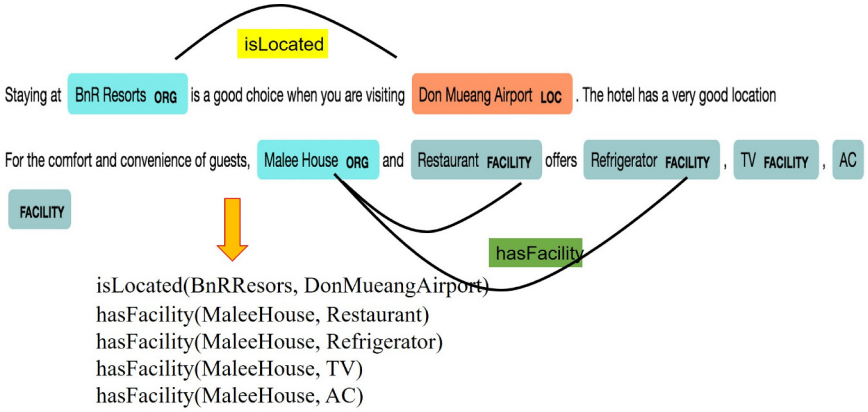


Fig.7: *Relation capturing part II.*

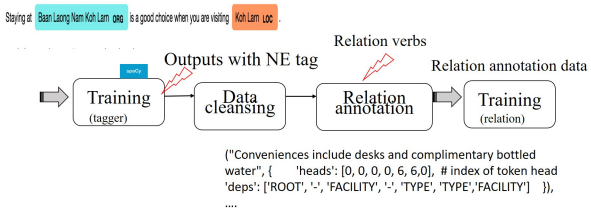


Fig.8: *Relation creation.*

	Sentence #	Word	POS	Tag
0	1	staying	VBG	O
1	1	at	IN	O
2	1	@	NN	B-ORG
3	1	Home	NN	I-ORG
4	1	Executive	NN	I-ORG
5	1	Apartment	NN	I-ORG
6	1	is	VBZ	O
7	1	a	DT	O
8	1	good	JJ	O

Fig.9: *BERT training data for NER*

4. EXPERIMENTAL RESULTS

The experiments were run on Intel 3.2 GHz Core i5 with 32 GB of RAM for training the models with 2 Titan Xps. We split the results into subsections: NER, and text classficiation for both SpaCy, and BERT for our tourism data set. All the code and data can be downloaded at `http://github.com/cchantra/nlp_tourism` for NER for all experimental code and data.

4.1 Name Entitiy Recognition (NER)

Using the data gathered in Section 3.1 we consider the NER task using SpaCy and BERT. The performance is reported below.

4.1.1 SpaCy

Three models were built using SpaCy. 1) the model to recognize ORG/LOC, 2) the model to recognize FACILITY, and 3) the model to recognize all ORG/LOC/FACILITY. For model 1), we used raw corpus with annotations only for ORG/LOC, containing 13,019 sentences. For 2), the raw corpus only contains FACILITY sentences, with 13,522 rows. For 3), we combined the corpus data from 1) and 2) to create one model recognizing all entities. Each model was trained until the loss was stable. Here, we manually select sentences for tagging.

The tagged data is divided into 70% training and 30% testing. Training for LOC/ORG had no difficulty at all since ORG/LOG labels are already in the pre-trained model of SpaCy. We added new words and labels and retrained it. For FACILITY, we added it as the new tag label and the model was trained to learn the new tags. The loss was higher than the LOC/ORG model. Also, when we combined the three tags, the total loss was even higher. If we do not manually select sentences containing these named entities, the large raw corpus takes a long time to train and the loss is very high, over 1,000.

Table 3: *Comparison between predicted and annotated NE using spaCy.*

Type	LOC/ORG		FAC		LOC/ORG/FAC	
	train	test	train	test	train	test
#Annotated	46,327	19,787	22,167	9,427	93,661	40,745
#Predicted	70,156	29,873	18,716	7,299	85,867	37387
Diff	23,829	10,086	-3,451	-2,128	-7,794	-3,358

Table 3 shows the correctness of the three models using the SpaCy approach. Row 'Annotated' shows the counts of the data labeled for training and 'Predicted' is the number of NEs predicted by the model. 'Diff' is the difference between'Annotated' and 'Predicted'. Note there are negative values in 'Diff'. This is because the model discovers more entities than the number of annotated labels. For LOC/ORG, SpaCy has the built-in (pre-trained) labels for it. Thus, it

may identify more entities than our number of annotated labels. For 'FACILITY', the number of predicted labels missed is around 15% for training and 22% for testing.

Note that SpaCy predicts the exact boundary of the phrases. The prediction results may not exactly match the annotation, but it provides the close boundary of the entity or good estimate. If we counted according to the number of exactly matched positions, it will be mostly wrong. Thus, we loosen the measurement to count only the number of predicted ones. For LOC/ORG/FAC, the number of missing ones is 8% for training and 8% for testing.

Consider the relation name together with the above name entities. Figure 10 shows the results of marking all NEs and relation names. Tag "USE" indicates "FACILITY", and "LOCATEAT" indicates "LOCATION". When creating one model for recognizing all five types of tag (ORG/LOC/FACILITY/USE/LOCATEAT), we obtain 95% accuracy for both training and testing.

Table 4 shows the accuracy results for five tags including relations. Row "Trained/annotated" shows the number of training annotated data for each tag. Row "Trained/predicted" is the number of predicted tags. Similarly, "Tested/annotated" is the number of tested annotated data entries. "Test/predicted" is the number of predicted tags. The average number of missing tags for both trained and test data is around 5%.

Table 4: *Annotated and Predicted Results for five tags.*

Data	LOC	USE	LOCATEAT	FACILITY	ORG
Train/ annotated	39,173	24,169	27,911	34,180	20,750
Train/ predicted	36,342	24,169	27,913	30,875	19,591
Test/ annotated	16,949	10,242	12,081	14,467	8,887
Test/predicted	15,716	10,245	12,084	13,053	8,459

Figure 11 presents an example of results. Figure 11 (a) is the prediction, while Figure 11 (b) is the actual annotation. If the phrases are correctly annotated like "Naga Peak Boutigue Resort", the prediction will be correct. For "Nong Thale", which is the name of location, the prediction includes the "city" word as well. "Ao Nang Krabi Boxing Stadium" is classified as ORG. SpaCy assumed the number 3.45 to be distance, so it is classified as LOC, but we do not label this in our dataset.

Figure 12 is another example where the manual annotation is not complete. With the pretrained model of SpaCy, it can suggest the entity types such as LOC, and ORG.

4.1.2 BERT

For BERT, we use the BIO tagging style. The training accuracy is depicted in Table 5, BERT performs well on all of the tasks. Original BERT (released November 2018) relies on its tokenizer which includes the token conversion to ID. It cannot tokenize our special multiword names which are proper nouns very well. For example, 'Central Pattaya' is tokenized into 'u'central', u'pat', u'##ta', u'##ya'. which causes our labels to have a wrong starting index at position 'pattaya'. The unknown proper nouns are chopped into portions, making the label positions shift out. According to the paper's suggestion, this needs to be solved by using own Wordpiece tokenizer [32]. In Table 5, we created our own set of vocabularies for training based on the data we crawled. The accuracy results are measured based on the number of correct predictions. LOC/ORG shows the results for location and name hotels. FAC shows the results for facilities of each hotels and the last column LOC/ORG/FAC shows the results for all tags together. The results for all cases are around 70%. The F1-score is quite low.

Table 6 shows precision, recall, and F1 -score of the last column (LOC/ORG/FAC) in Table 5, measured by *sklearn-metric*. The recall values are low.

Table 5: *Comparison between predicted and annotated NE using BERT with own vocabulary.*

Type	LOC/ORG		FAC		LOC/ORG/FAC	
	train	test	train	test	train	test
Loss	0.019	0.0165	0.375	0.029	0.123	0.109
Accuracy	0.751	0.717	0.859	0.7484	0.736	0.629
F1	0.258	0.346	0.245	0.245	0.287	0.464

Table 6: *Precision, Recall, F1 for BERT with own vocabulary.*

Tag	precision	recall	F1-score	support
B-ORG	1.00	0.08	0.15	50024
I-ORG	1.00	0.20	0.33	39288
B-LOC	0.99	0.12	0.21	22723
I-LOC	1.00	0.08	0.15	13782
B-FAC	1.00	0.27	0.43	37335
I-FAC	0.99	0.54	0.70	13672
O	0.53	1.00	0.69	163016

If we consider using the default BERT tokenizer, ignoring the tokenization problem mentioned previously, we see in Table 7, the accuracy, precision, recall, and F1-score are higher.

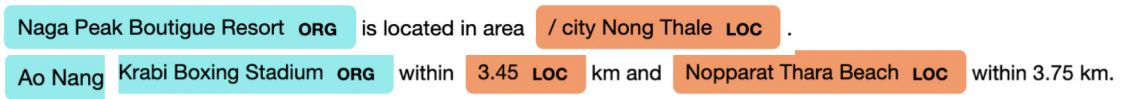
Table 8 shows precision, recall, and F1-score. Compared to Table 6, the recall is higher in most cases, and thus the F1-score is higher.

Table 7: *Comparison between predicted and annotated NE using BERT (wordpiece).*

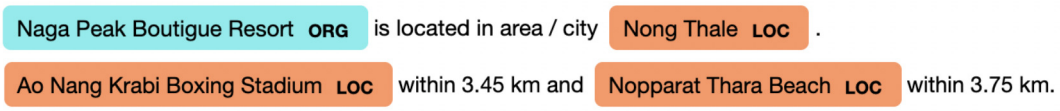
Type	LOC/ORG		FAC		LOC/ORG/FAC	
	train	test	train	test	train	test
Loss	0.061	0.153	0.03	0.0295	0.123	0.109
Accuracy	0.958	0.955	0.939	0.932	0.866	0.866
F1	0.737	0.715	0.421	0.431	0.442	0.464

Both tables are based on the BERT pre-training model which is BERT-Large, Uncased (Whole Word Masking): 24-layer, 1024-hidden, 16-heads, 340M parameters. In the default pre-processing code, Word-

Fig.10: *LOC/ORG/FACILITY/Relation marking*



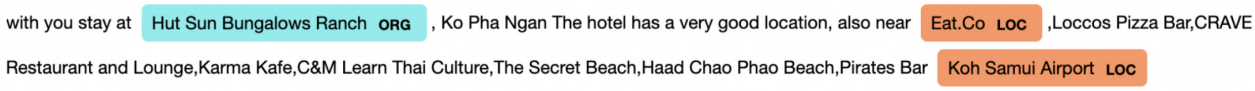
(a)



(b)

Fig.11: *Example results part I.*

(a)



(b)

Fig.12: *Example results part II.*

Table 8: *Precision, Recall, F1 for BERT with default tokenization (wordpiece).*

Tag	precision	recall	F1-score	support
B-ORG	0.87	0.33	0.48	30821
I-ORG	0.79	0.50	0.61	37809
B-LOC	0.98	0.37	0.54	21445
I-LOC	0.99	0.23	0.38	14321
B-FAC	0.86	0.54	0.67	49396
I-FAC	0.99	0.33	0.49	67105
O	0.85	0.99	0.91	734663

Piece tokens are randomly selected to mask. The new technique is called Whole Word Masking. In this case, all of the the tokens corresponding to a word are masked at once. The overall masking rate remains the same. The training is identical – we can still predict each masked WordPiece token independently. In Table 8, the improvement comes from the higher recall, while the precision is lower due to the default BERT vocabularies.

4.1.3 BERT on other data sets

Still, the F1-score is not quite satisfactory. This is likely due to the vocabularies and keywords as well as the data set. Also, there is another problem, which is that the data is not cleaned enough. We retry the experiments on other similar kinds of data sets with clear keywords. In the second data set, as mentioned in Section 3.1, we extracted only the review comments from TripAdvisor. There are three categories of reviews: hotel review, shop and tourism review, and restaurant review. We perform the same preprocessing as in the previous experiment in Section 4.1.2. The keywords are built based on the review types. Each data set has the characteristics in Table 9. Under Column “size”, “original” is the total count of data lines scraped. “cleaning” shows the number of data elements which remained

Table 9: *Data set for restaurant, shopping, and hotel review*

	size		Total size		category and size	
Dataset	original	cleaning	train	test	train	test
Restaurant	18,700	10,010	5,399	4,611	location(0):1,526 food(1):3,873	location(0):1,312 food(1):3,299
Hotels	11,859	10,447	6,272	4,175	location(0):3,300 service(1):2,972	location(0):2,528 service(1):1,647
Shopping&Tourism	12,253	9,162	5,720	3,442	nature-travel(0):2,958 shopping-travel(1):2,762	nature-travel(0):2,200 shopping-travel(1):1,242

after cleansing. The cleansing and preprocessing processes do things such as removing irrelevant columns, creating a list of keywords based on types (location, service, etc.), maintaining the data rows containing the keywords, and finding the category type (label) of each row. The keywords are simpler than in the previous cases since they are only indicators in the reviews that imply the locations or services. They are not complicated proper nouns like in the previous data set. For example, location keywords are beach side, street food, jomtien beach, kantary beach, next to, etc. The service keywords are cleaning, bathroom, swimming pool, shower, amentities, etc. Finally, we save the selected data rows containing keywords in a new file. The columns "train" and "test" show the data size divided into training and testing sets. Column "category and size" shows the category for each type of data and size. The data is used for other tasks such as the text classification task.

Table 10 presents the F1-score, validation loss, and validation accuracy of BERT-NER task on each row of Table 9. The table shows higher values for all data sets. For Shopping & Tourism and Restaurant data sets, it yields the highest accuracy. This confirms the effectives of the approach when a cleaner data set is used.

Table 10: *BERT-NER tasks for hotels, shopping, and restaurant data sets.*

Data set	F1-score	Val loss	Val accuracy
Hotels	0.7796	0.0145	0.9933
Shopping & Tourism	0.9888	0.0037	0.9997
Restaurant	0.9839	0.0052	0.9986

Tables 11-13 presents the detail score of each category. In Table 11 RLOC and RSER show the BIO tag for location and service tags in Table 9. In Table 12, NTV and STV are the BIO tags for natural-travel and shopping travel respectively and in Table 13, FOOD and RLOC are the BIO tags for food and location tags respectively. For all of the tags, we get high score for all precision, recall and F1-score. This shows the effectiveness of the BERT-NER in the data set.

4.1.4 Comparison to other NERs

Table 14 compares our approach with other approaches. The model types are different, and different tags are considered. Compared to our framework, we focus on the methodology of automated data preparation steps and use pre-trained network models as a tool.

Table 11: *Precision, Recall, and F1-Score for BERT Hotel.*

Tag	precision	recall	F1-score	support
B-RLOC	0.98	0.97	0.97	829
I-RLOC	0.99	0.99	0.99	792
B-RSER	0.96	0.93	0.95	133
I-RSER	1.00	1.00	1.00	15
O	1.00	1.00	1.00	131991

Table 12: *Precision, Recall, F1-score for BERT Shopping & Tourism.*

Tag	precision	recall	F1-score	support
B-NTV	0.98	0.99	0.99	867
I-NTV	0.97	0.98	0.98	539
B-STV	0.82	0.66	0.73	35
I-STV	0.71	0.48	0.57	25
O	1.00	1.00	1.00	112454

Table 13: *Precision, Recall, F1-score for BERT Restaurant.*

Tag	precision	recall	F1-score	support
B-FOOD	0.99	0.98	0.99	1888
I-FOOD	0.97	0.99	0.98	449
B-RLOC	0.99	0.99	0.99	469
I-RLOC	0.97	0.98	0.97	122
O	1.00	1.00	1.00	168336

Table 14: *Comparison between predicted and annotated NE using BERT.*

Approach	type	model	data set	tourism tag
Vijay et.al.[33]	Supervised	CRF	TripAdvisor, Wikipedia	hotel name, POI, review, star,...
Saputro et.al.[34]	Semi-Supervised	Naïve Bayes,SSL	Top google search	nature, region, place, city, negative
Our work	Supervised	SpaCy,BERT	TripAdvisor, Traveloka, Hotels.com	Location, Facility, Organization

4.2 Text Classification

We use the review data set in three categories. For each category, in labeling, we use a category number, representing the category of each review. For example, for restaurant data set, the label of 0 means the review is about location, and the label of 1 means the review is about food. For the classification experiment, we used numbered labels. For example, "50m to the beach means you'll reach the sea within that distance" is in "Location Reviews", which is the

category numbered 0.

4.2.1 SpaCy

Figure 13 shows the accuracy, precision, recall, and F1-score of the Restaurant data set in Table 9 for text classification task. Each graph uses different sized vector representations for embedding,e.g., 1,000, 500, 200, 50. In the figure, X-axis is the iteration number. Most of the sizes give good performance. For this data set, the vector size 200 yields better performance than the others. Similarly, Figure 14 shows the accuracy, precision, recall, and F1-score of the Shopping&Tour data set where the vector size 1,000 yields better performance than others, i.e. around 0.99. Figure 15 shows the accuracy, precision, recall, and F1-score of the Hotel data set in Table 9. For this data set, the vector size 1,000 yields better performance than the others, i.e. around 0.99.

4.2.2 BERT

For the classification purpose, the classification layer is added on top of the BERT output layer. The default BERT tokenizer is used. For each data set, the category type values in Table 9 are used as labels for training. We follow the default implementation in [2]. Note that BERT can be used for text summarization, i.e, BERT as an extractive summarization [35] with our data. The BERT layers are modified to accommodate in the sentence level, called BERTSum. Next, the summarization layer is added on top of BERT output layers (BERTSum) [36]. The summarization layer can be any one of three types: simple, transformer, and LSTM.

Table 15 shows the performance of two classification tasks. Classification task (1) uses the linear layer on top of BERT layers. Classification task (2) allows a linear classifier followed by a sigmoid function. The results are shown Table 15. The performance on the hotel data set is the best for both classification tasks in accuracy, precision, recall, and F1-score.

Table 15: *BERT for a classification task.*

Type	Eval Acc	Loss	Precision	Recall	F1-score
Classification task (1)					
Hotels	0.9930	0.0271	0.9855	0.9969	0.9912
Shop-tour	0.9866	0.061	0.9791	0.9838	0.9815
Restaurant	0.9514	0.2484	0.9635	0.9687	0.9661
Classification task (2)					
Hotels	0.9770	0.3375	0.9543	0.9891	0.9713
Shop-Tour	0.9686	0.3451	0.9388	0.9766	0.9573
Restaurant	0.9255	0.3886	0.9443	0.9521	0.9482

4.3 Comparison to all tasks

We compared the results from SpaCy to those from BERT for the three datasets on 4 tasks. We compared three metrics, including accuracy, and F1-score, in Figure 16. BERT performs well for all tasks, while SpaCy classification yields a bit less accuracy than BERT's.

5. CONCLUSION AND FUTURE WORK

This paper presents a methodology for extracting information from tourism data. In particular, we demonstrate prototypes of location and facility extraction of hotel accommodations using machine learning. We first constructed the data set using WWW crawling of tourism websites. Then, we presented the preprocessing methodology which builds the specific vocabularies, raw corpus, and annotations necessary for the model construction. Then, two model frameworks, BERT and SpaCy were used to construct the models to perform tasks on the data set: recognizing name entities and text classification. The performance of the two approaches was discussed. The accuracy and F1-score on BERT depend on the characteristics of the data set, i.e., how clean the data set is, and the annotation method. The accuracy of BERT is a little higher than that of SpaCy for a clean data set. The methodology can be generalized to recognize entities of other domains to extract relevant information such as facilities for medical care and the review opinions. Other machine learning models could be for preprocessing the training data.

References

[1] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: `http://arxiv.org/abs/1810.04805`

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018. [Online]. Available: `https://github.com/google-research/bert`

[3] C. Chantrapornchai and C. Choksuchat, "Ontology construction and application in practice case study of health tourism in thailand," *SpringerPlus*, vol. 5, no. 1, p. 2106, Dec 2016. [Online]. Available: `https://doi.org/10.1186/s40064-016-3747-3`

[4] C. Feilmayr, S. Parzer, and B. Pröll, "Ontology-based information extraction from tourism websites," J. of IT & Tourism, vol. 11, pp. 183-196, 08 2009.

[5] H. Alani, , D. E. Millard, M. J. Weal, W. Hall, P. H. Lewis, and N. R. Shadbolt, "Automatic ontology-based knowledge extraction from web documents," *IEEE Intelligent Systems*, vol. 18, no. 1, pp. 14-21, Jan 2003.

[6] R. Jakkilinki, M. Georgievski, and N. Sharda, "Connecting destinations with an ontologybased e-tourism planner," in *Information and Communication Technologies in Tourism 2007*, M. Sigala, L. Mich, and J. Murphy, Eds. Vienna: Springer Vienna, 2007, pp. 21-32.

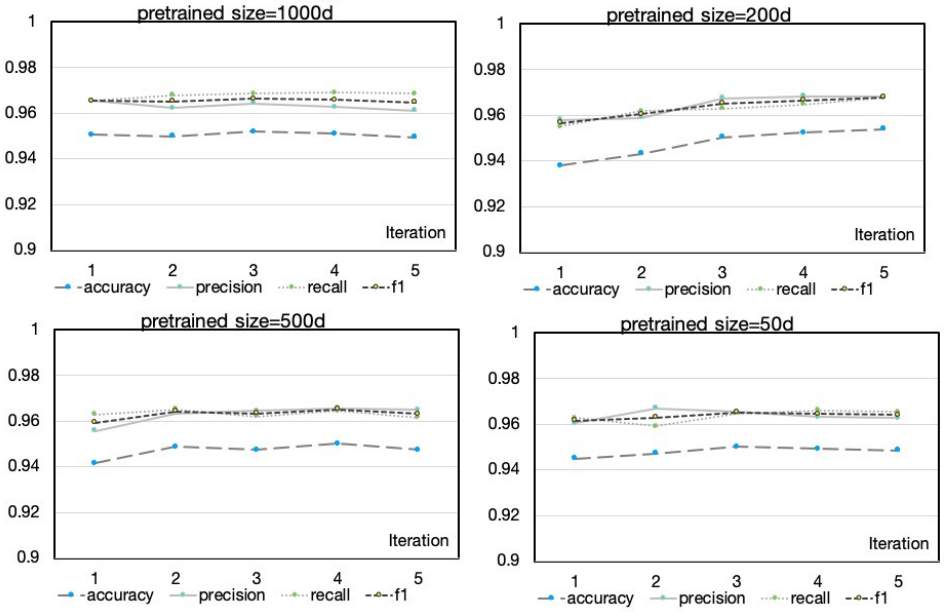


Fig.13: *Accuracy, precision, recall, and F1-score of Restaurant dataset.*

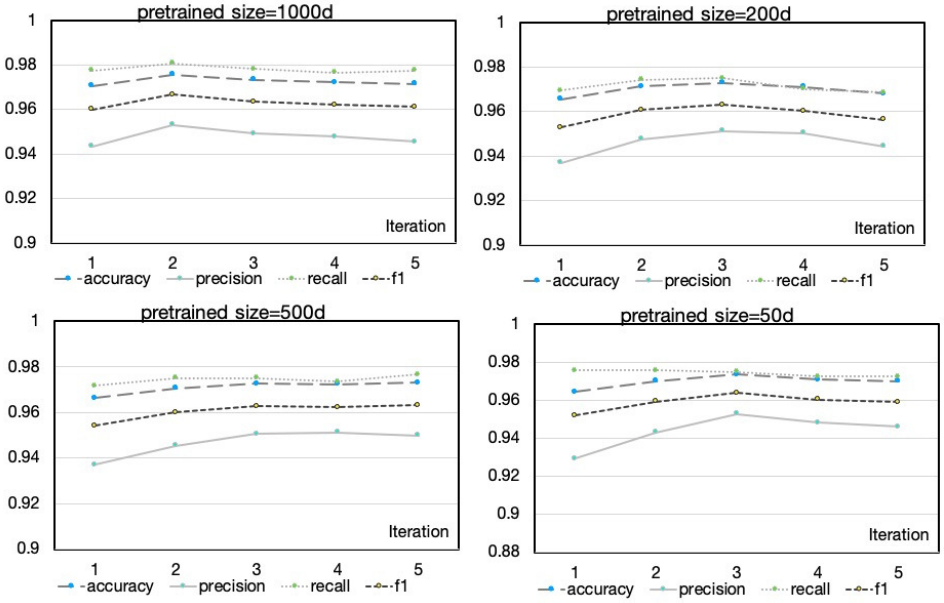


Fig.14: *Accuracy, precision, recall, and F1-score of Shopping & Tour dataset.*

[7] S. Mouhim, A. Aoufi et al., "A knowledge management approach based on ontologies: the case of tourism," *SpringerPlus*, vol. 4, no. 3, pp.362-369, 2011.

[8] D. Bachlechner. (2004) OnTour The Semantic Web and its Benefits to the Tourism Industry. Retrieved 13 June 2019. [Online]. Available: `https://pdfs.semanticscholar.org/eabc/d4368f5b00e248477a67d710c058f46cd83e.pdf`

[9] M. Sigala, L. Mich et al., "Connecting destinations with an ontology-based e-tourism planner," in *Information and communication technologies in tourism*, M. Sigala and M. L. Mich, Eds. Springer, Vienna, 2007, pp. 21-32.

[10] STI INNBUCK. (2013) Accommodation Ontology Language Reference. Retrieved 8 March 2019. [Online]. Available: `http://ontologies.sti-innsbruck.at/acco/ns.html`

[11] Web vocabulary for e-commerce. (2008) A paradigm shift for e-commerce. Since 2008. Retrieved 8 March 2019. [Online]. Available: `http://www.heppnetz.de/projects/goodrelations/`

[12] M. Chaves, L. Freitas, and R. Vieira, "Hontology: A multilingual ontology for the accommoda-

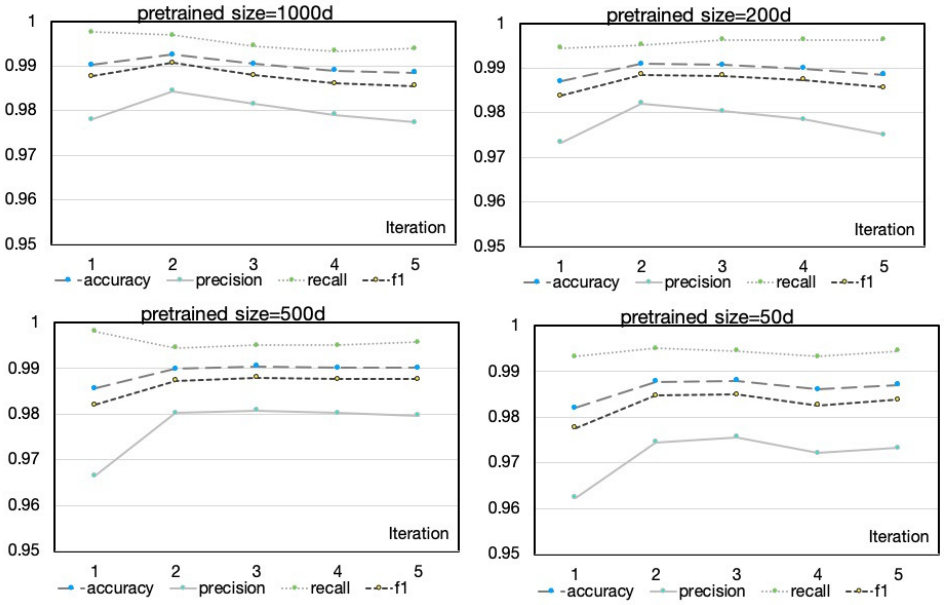


Fig.15: *Accuracy, precision, recall, and F1-score of Hotel dataset.*

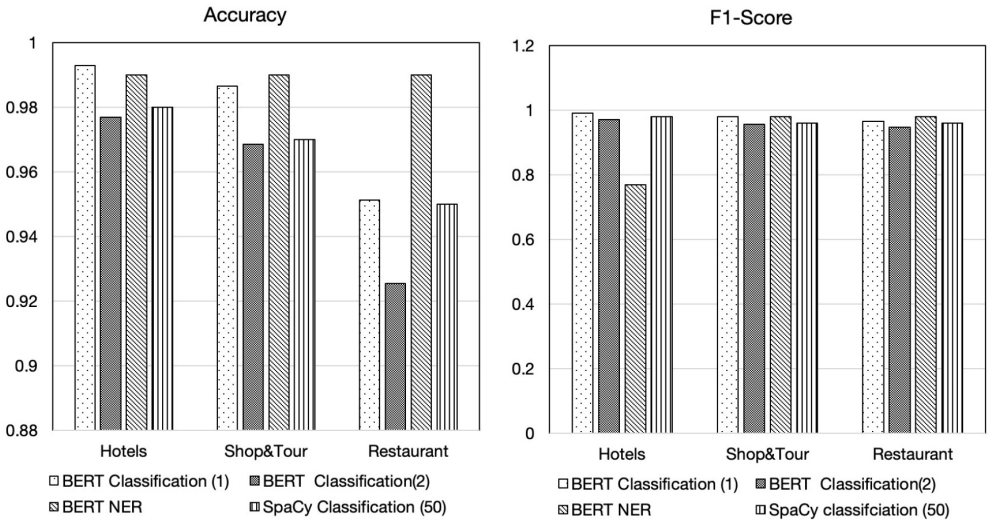


Fig.16: *Comparison all tasks.*

tion sector in the tourism industry," 01 2012.

[13] K. Vila and A. Ferrández, "Developing an ontology for improving question answering in the agricultural domain," in *Metadata and Semantic Research*, F. Sartori, M. Á. Sicilia, and N. Manouselis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 245-256.

[14] W. Khan, A. Daud, J. Nasir, and T. Amjad, "A survey on the state-of-the-art machine learning models in the context of nlp," vol. 43, pp. 95-113, 10 2016.

[15] SpaCy. (2020) Facts & Figures. Retrieved 26 March 2020. [Online]. Available: `https://spacy.io/usage/facts-figures`

[16] ———, "Library architecture," 2019. [Online]. Available: `https://spacy.io/api#section-nn-model`

[17] ———. (2020) Facts & Figures. Retrieved 26 March 2020. [Online]. Available: `https://spacy.io/usage/linguistic-features`

[18] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning in natural language processing," *CoRR*, vol. abs/1807.10854, 2018. [Online]. Available: `http://arxiv.org/abs/1807.10854`

[19] T. Shi, Y. Keneshloo, N. Ramakrishnan, and C. K. Reddy, "Neural abstractive text summarization with sequence-to-sequence models," *CoRR*, vol. abs/1812.02303, 2018. [Online]. Available: `http://arxiv.org/abs/1812.02303`

[20] L. Yang,"Abstractive summarization for amazon reviews," Stanford University, CA, Tech. Rep.

[Online]. Available: https://cs224d.stanford.edu/reports/lucilley.pdf

[21] J. C. Cheung, "Comparing abstractive and extractive summarization of evaluative text: Controversiality and content selection," Canada, 2008. [Online]. Available: http://www.cs.toronto.edu/$\sim$jcheung/papers/honours-thesis.pdf

[22] Y. Liu and M. Lapata, "Text summarization with pretrained encoders," in *The Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, 2019, pp. 3730-3740.

[23] D. Miller, "Leveraging BERT for extractive text summarization on lectures," *CoRR*, vol. abs/1906.04165, 2019. [Online]. Available: http://arxiv.org/abs/1906.04165

[24] M. Munikar, S. Shakya, and A. Shrestha, "Fine-grained sentiment classification using BERT," in *2019 Artificial Intelligence for Transforming Business and Society (AITB)*, vol. 1, 2019, pp. 1-5.

[25] A. Adhikari, A. Ram, R. Tang, and J. Lin, "DocBERT: BERT for document classification," *ArXiv*, vol. abs/1904.08398, 2019.

[26] X. Dai, "Recognizing complex entity mentions: A review and future directions," in *Proceedings of ACL 2018, Student Research Workshop*, Melbourne, Australia, 2018, pp. 37-44.

[27] V. Yadav and S. Bethard, "A survey on recent advances in named entity recognition from deep learning models," in *Proceedings of 27th International Conference on Computational Linguistics*, Santa Fe, New Mexico, USA, 2018, pp. 2145-2158.

[28] M. Zhu, Z. Deng, W. Xiong, M. Yu, M. Zhang, and W. Y. Wang, "Towards open-domain named entity recognition via neural correction models," 2019.

[29] F. Souza, R. Nogueira, and R. Lotufo, "Portuguese named entity recognition using bert-crf," 2019.

[30] K. Xue, Y. Zhou, Z. Ma, T. Ruan, H. Zhang, and P. He, "Fine-tuning bert for joint entity and relation extraction in chinese medical text," in *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2019, pp. 892-897.

[31] "NLP-progress." [Online]. Available: http://nlpprogress.com/english/namedentityrecognition.html

[32] Y. Wu et al., "Google's neural machine translation system: Bridging the gap between human and machine translation," CoRR, vol. abs/1609.08144, 2016. [Online]. Available: http://arxiv.org/abs/1609.08144

[33] J. Vijay and R. Sridhar, "A machine learning approach to named entity recognition for the travel and tourism domain," *Asian Journal of Information Technology*, vol. 15, pp. 4309-4317, 01 2016.

[34] K. E. Saputro, S. S. Kusumawardani, and S. Fauziati, "Development of semi-supervised named entity recognition to discover new tourism places," *2016 2nd International Conference on Science and Technology-Computer (ICST)*, pp. 124-128, 2016.

[35] Y. Liu, "Fine-tune BERT for extractive summarization," *CoRR*, vol. abs/1903.10318, 2019. [Online]. Available: http://arxiv.org/abs/1903.10318

[36] A. Vashisht, "BERT for text summarization," 2019. [Online]. Available: https://iq.opengenus.org/bert-for-text-summarization/

Chantana Chantrapornchai received the bachelor's degree in computer science from Thammasat University of Thailand, in 1991, the master's degree from Northeastern University at Boston, College of Computer Science, in 1993 and the PhD degree from the University of Notre Dame, Department of Computer Science and Engineering, in 1999. Currently, she is an associated professor with the Department of Computer Engineering, Faculty of Engineering, Kasetsart University, Thailand. Her research interests include parallel computing, big data processing, deep learning application, semantic web, computer architecture, and fuzzy logic. She is also affiliated with HPCNC laboratory. She is currently a principle investigator of GPU Education program and NVIDIA DLI Ambassador at Kasetsart University.

Aphisit Tunsakul received the Bachelor's degree in Computer Engineering from Mea Fah Luang University of Thailand, in 2014, and Master's degree from Kasetsart University at Thailand, College of Computer Engineering, in 2019 respectively. Currently, he is a researcher affiliated with the Department of Computer Vision, Faculty of National Security and Dual-Use Technology Center: NSD, National Science and Technology Development Agency. His research interests include Computer vision, Natural language Processing and Big data. He is currently with NSD laboratory at National Science and Technology Development Agency.