ETNLP: A Visual-Aided Systematic Approach to Select Pre-Trained Embeddings for a Downstream Task

Xuan-Son Vu[1], **Thanh Vu**[2], **Son N. Tran**[3], **Lili Jiang**[1]
[1]Umeå University, Sweden
[2]The Australian E-Health Research Centre, CSIRO, Australia
[3] The University of Tasmania, Australia;
{sonvx, lili.jiang}@cs.umu.se
thanh.vu@csiro.au, sn.tran@utas.edu.au ;

Abstract

Given many recent advanced embedding models, selecting pre-trained word embedding (a.k.a., word representation) models best fit for a specific downstream task is non-trivial. In this paper, we propose a systematic approach, called *ETNLP*, for extracting, evaluating, and visualizing multiple sets of pre-trained word embeddings to determine which embeddings should be used in a downstream task.

We demonstrate the effectiveness of the proposed approach on our pre-trained word embedding models in Vietnamese to select which models are suitable for a named entity recognition (NER) task. Specifically, we create a large Vietnamese word analogy list to evaluate and select the pre-trained embedding models for the task. We then utilize the selected embeddings for the NER task and achieve the new state-of-the-art results on the task benchmark dataset. We also apply the approach to another downstream task of privacy-guaranteed embedding selection, and show that it helps users quickly select the most suitable embeddings. In addition, we create an open-source system using the proposed systematic approach to facilitate similar studies on other NLP tasks. The source code and data are available at https://github.com/vietnlp/etnlp.

1 Introduction

Word embedding, also known as word representation, represents a word as a vector capturing both syntactic and semantic information, so that the words with similar meanings should have similar vectors (Levy and Goldberg, 2014). Although, the classical embedding models, such as Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), fastText (Bojanowski et al., 2017), have been shown to help improve the performance of existing models in a variety of Natural Language Processing (NLP) tasks like parsing (Bansal et al., 2014), topic modeling (Nguyen et al., 2015), and document classification (Taddy, 2015; Vu et al., 2018b). Each word is associated with a single vector leading to a challenge on using the vector across linguistic contexts (Peters et al., 2018). To handle the problem, recently, contextual embeddings (e.g., ELMO of Peters et al. (2018), BERT of Devlin et al. (2018)) have been proposed and help existing models achieve new state-of-the-art results on many NLP tasks. Different from non-contextual embeddings, ELMO and BERT can capture different latent syntactic-semantic information of the same word based on its contextual uses. Therefore, for completeness, in this paper, we incorporate both classical embeddings (i.e., Word2Vec, fastText) and contextual embeddings (i.e., ELMO, BERT) to evaluate their performances on NLP downstream tasks.

Given the fact that there are many different types of word embedding models, we argue that having a systematic pipeline to evaluate, extract, and visualize word embeddings for a downstream NLP task, is important but non-trivial. However, to our knowledge, there is no single comprehensive pipeline (or toolkit) which can perform all the tasks of evaluation, extraction, and visualization. For example, the recent framework called *flair* (Akbik et al., 2018) is used for training and stacking multiple embeddings but does not provide the whole pipeline of extraction, evaluation and visualization.

In this paper, we propose *ETNLP*, a systematic pipeline to extract, evaluate and visualize the pre-trained embeddings on a specific downstream NLP task (hereafter ETNLP pipeline). The ETNLP pipeline consists of three main components which are *extractor*, *evaluator*, and *visualizer*. Based on the vocabulary set within a downstream task, the extractor will extract a subset of word embeddings for the set to run evaluation

Proceedings of Recent Advances in Natural Language Processing, pages 1285–1294,
Varna, Bulgaria, Sep 2–4, 2019.
https://doi.org/10.26615/978-954-452-056-4_147

and visualization. The results from both *evaluator* and *visualizer* will help researchers quickly select which embedding models should be used for the downstream NLP task. On the one hand, the *evaluator* gives a concrete comparison between multiple sets of word embeddings. While, on the other hand, the *visualizer* will give the sense on what type of information each set of embeddings preserves given the constraint of the vocabulary size of the downstream task. We detail the three main components as follows.

- **Extractor** extracts a subset of pre-trained embeddings based on the vocabulary size of a downstream task. Moreover, given multiple sets of pre-trained embeddings, how do we get the advantage from a few or all of them? For instance, if people want to use the character embedding to handle the out-of-vocabulary (OOV) problem in Word2Vec model, they have to implement their own extractor to combine two different sets of embeddings. It is more complicated when they want to evaluate the performance of either each set of embeddings separately or the combination of the two sets. The provided **extractor** module in ETNLP will fulfill those needs seamlessly to elaborate this process in NLP applications.

- **Evaluator** evaluates the pre-trained embeddings for a downstream task. Specifically, given multiple sets of pre-trained embeddings, how do we choose the embeddings which will potentially work best for a specific downstream task (e.g., NER)? Mikolov et al. (2013) presented a large benchmark for embedding evaluation based on a series of analogies. However, the benchmark is only for English and there is no publicly available *large* benchmark for low resource languages like Vietnamese (Vu et al., 2014). Therefore, we propose a new evaluation metric for the word analogy task in Section 3.

- **Visualizer** visualizes the embedding space of multiple sets of word embeddings. When having a new set of word embeddings, we need to get a sense of what kinds of information (e.g., syntactic or semantic) the model does preserve. We specifically want to get samples from the embedding set to see what is the semantic similarity between different words. To fulfill this requirement, we design two different visualization strategies to explore the embedding space: (1) side-by-side visualization and (2) interactive visualization.

The side-by-side visualization helps users compare the qualities of the word similarity list between multiple embeddings (see figure 5). It allows researchers to "zoom-out" and see at the overview level what is the main difference between multiple embeddings. Moreover, it can visualize large embeddings up to the memory size of the running system. Regarding implementation, we implemented this visualization from scratch running on a lightweight webserver called Flask (`flask.pocoo.org`).

For the interactive visualization, it helps researchers "zoom-in" each embedding space to explore how each word is similar to the others. To do this, the well-known Embedding Projector (`projector.tensorflow.org`) is employed to explore the embedding space interactively. Unlike the side-by-side visualization, this interactive visualization can only visualize up to a certain amount of embedding vectors as long as the tensor graph is less than 2GB. This is a big limitation of the interactive visualization approach, which we plan to improve in the near future. Finally, it is worth to mention that the visualization module is dynamic and it does not require to change any codes when users want to visualize multiple pre-trained word embeddings.

To demonstrate the effectiveness of the ETNLP pipeline, we employ it to a use case in Vietnamese. Evaluating pre-trained embeddings in Vietnamese is a challenge as there is no publicly available *large*[1] lexical resource similar to the word analogy list in English to evaluate the performance of pre-trained embeddings. Moreover, different from English where all word analogy records consist of a single syllable in one record (e.g., grandfather | grandmother | king | queen), in Vietnamese, there are many cases where only words formulated by multiple syllables can represent a word analogy record (e.g., ông nội | bà ngoại | vua | nữ_hoàng).

We propose a large word analogy list in Vietnamese which can handle the problems. Having that word analogy list constructed, we utilize different embedding models, namely Word2Vec, fastText, ELMO and BERT on Vietnamese Wikipedia data to generate different sets of word embeddings. We then utilize the word analogy list to select suitable sets of embeddings for the named entity recognition (NER) task in Vietnamese. We achieve

[1] There are a couple of available datasets (Nguyen et al., 2018b). But the datasets are small containing only 400 words.

the new state-of-the-art results on VLSP 2016[2], a Vietnamese benchmark dataset for the NER task.

Here are our key contributions in this work:

- Propose a systematic pipeline (ETNLP) to evaluate, extract, and visualize multiple sets of word embeddings on a downstream task.

- Release a large word analogy list in Vietnamese for evaluating multiple word embeddings.

- Train and release multiple sets of word embeddings for NLP tasks in Vietnamese, wherein, their effectiveness is verified through new state-of-the-art results on a NER task in Vietnamese.

The rest of this paper is organized as follows. Section 2 describes how different embedding models are trained. Section 3 shows how to use ETNLP to extract, evaluate, and visualize word embeddings. Section 4 explains how the word embeddings are selected for the NER task using the word analogy task. Section 5 concludes the paper followed by future work.

2 Embedding Models

This section details the word embedding models incorporated in our systematic pipeline.

- **Word2Vec (W2V)** (Mikolov et al., 2013): a widely used method in NLP for generating word embeddings.

- **W2V_C2V**: the Word2Vec (W2V) model faces the OOV issue on unseen text, therefore, we provide a character2vec (C2V) (Kim et al., 2015) embedding for unseen words. When the C2V is not available, it can be easily calculated from a W2V model by averaging all vectors where a character occurred. Our experiments further confirm this averaging approach is efficient.

- **fastText** (Bojanowski et al., 2016): it associates embeddings with character-based n-grams, and a word is represented as the summation of the representations of its character-based n-grams. Based on this design, fastText attempts to capture morphological information to induce word embeddings, and hence, deals better with OOV words.

- **ELMO** (Peters et al., 2018): a model generates embeddings for a word based on the context it appears. Thus, we choose the contexts where the word appears in the training corpus to generate embeddings for each of its occurrences. Then the final embedding vector is the average of all its context embeddings.

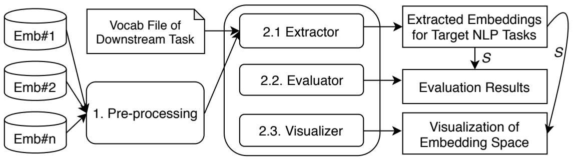

Emb#1
Emb#2
Emb#n
Vocab File of Downstream Task
1. Pre-processing
2.1 Extractor
2.2. Evaluator
2.3. Visualizer
Extracted Embeddings for Target NLP Tasks
S
S
Evaluation Results
Visualization of Embedding Space

Figure 1: General process of the ETNLP pipeline where $\mathcal{S}$ is the set of extracted embeddings for Evaluation and Visualization of multiple embeddings on a downstream NLP task.

- **BERT_{Base, Large}** (Devlin et al., 2018): BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. Different from ELMO, the directional models, which reads the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words simultaneously. It, therefore, is considered bidirectional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word). BERT comes with two configurations called BERT_Base (12 layers) and BERT_Large (24 layers). To get the embedding vector of a word, we average all vectors of its subwords. Regarding contexts, similar to the ELMO model above, we choose the contexts where the word appears in the training corpus.

3 Systematic Pipeline

Figure 1 shows the general process of the ETNLP pipeline. The four main processes of ETNLP are very simple to call from either the command-line or the Python API.

- **Pre-processing:** since we use Word2Vec (W2V) format as the standard format for the whole process of ETNLP, we provide a pre-processing tool for converting different embedding formats to the W2V format.

- **Extractor:** to extract embedding vectors at word level for the specific target NLP task (i.e., NER task in our case). For instance, the popular implementation of Reimers and Gurevych (2017) on the sequence tagging task allows users to set location for the word embeddings. The format of the

[2] http://vlsp.org.vn/vlsp2016/eval/ner

```
$python3 etnlp_api.py  -input     "<emb_in#1>;<emb_in#2>"
                       -input_c2v <emb_in#3>
                       -vocab <file>
                       -output <out_file.gz>
                       -args extract;solveoov:1
```

Figure 2: Run *extractor* to export single or multiple embeddings for NLP tasks.

file is text-based, i.e., each line contains the embedding of a word. The file then is compressed in .gz format. Figure 2 shows a command-line to extract multiple embeddings for an NLP task. The argument "-vocab" is the location to a vocabulary list of the target NLP task (i.e., the NER task) which is extracted from the task training data. The option "solveoov:1" informs the *extractor* to use Character2Vec (C2V) embedding to solve OOV words in the first embedding "<emb_in#1>". The "-input_c2v" can be omitted if users wish to simply extract embeddings from the embedding list given after the "-input_embs" argument. Output of this phase is a set of embeddings $\mathcal{S}$ to run on the next evaluation phase.

- **Evaluator** evaluates multiple sets of embeddings (i.e., $\mathcal{S}$) on the word analogy task. Based on the performance of each set of embeddings in $\mathcal{S}$, we can decide what embeddings are used in the target NLP task. To do this evaluation, users have to set the location of the word embeddings and the word analogy list. For more convenience to represent the compound words, we use " | " to separate different part of a word analogy record instead of space as in the English word analogy list. Figure 3 shows an example of two records in the word analogy in Vietnamese (on the left) and their translation (on the right). The lower part shows a command-line to evaluate multiple sets of word embeddings on this task. Regarding this *evaluator*, it is worth to note that with a huge number of possible linguistic relations (and different objectives, e.g., modeling syntactic vs. semantic properties), no embedding model is able to hold all related words close in the vector space. Therefore, only one testing schema (i.e., word analogy test) is not enough to evaluate multiple pre-trained embeddings. Thus, ETNLP is designed with the capability to be easily plugged in more tests, which makes *evaluator* more robust. However, in this paper, our experimental results showed that, word analogy task is sufficient to select good embeddings for the NER task in Vietnamese.

Vietnamese	English
ông nội | bà ngoại | ông | bà	grandfather | grandmother | grandpa | grandma
ông nội | bà ngoại | vua | nữ_hoàng	grandfather | grandmother | king | queen

```
$python3 etnlp_api.py  -input "<emb_in#1>;<emb_in#2>"
                       -analoglist <file>
                       -output <eval_results> -args eval
```

Figure 3: Run *evaluator* on multiple word embeddings on the word analogy task.

```
$python3 etnlp_api.py  -input     "<emb_in#1>;<emb_in#2>"
                       -args visualizer
```

Figure 4: Run *visualizer* to explore given pre-trained embedding models.

- **Visualizer:** to visualize given word embeddings in the argument "-input_embs" in both zoom-out (the side-by-side visualization) and zoom-in (the interactive visualization) manners. For the zoom-out, users type a word that they want to compare the similar words in different embedding models (see Figure 5). For the zoom-in, after the executions, embedding vectors are transformed to tensors to visualize with the Embedding Projector. Each word embedding will be set to different local port from which, users can explore the embedding space using a Web browser. Figure 6 shows an example of the interactive visualization of "Hà_Nội"$_{Hanoi}$ using ELMO embeddings. See Figure 4 for an example command-line.

4 Evaluations: A Use-Case in Vietnamese

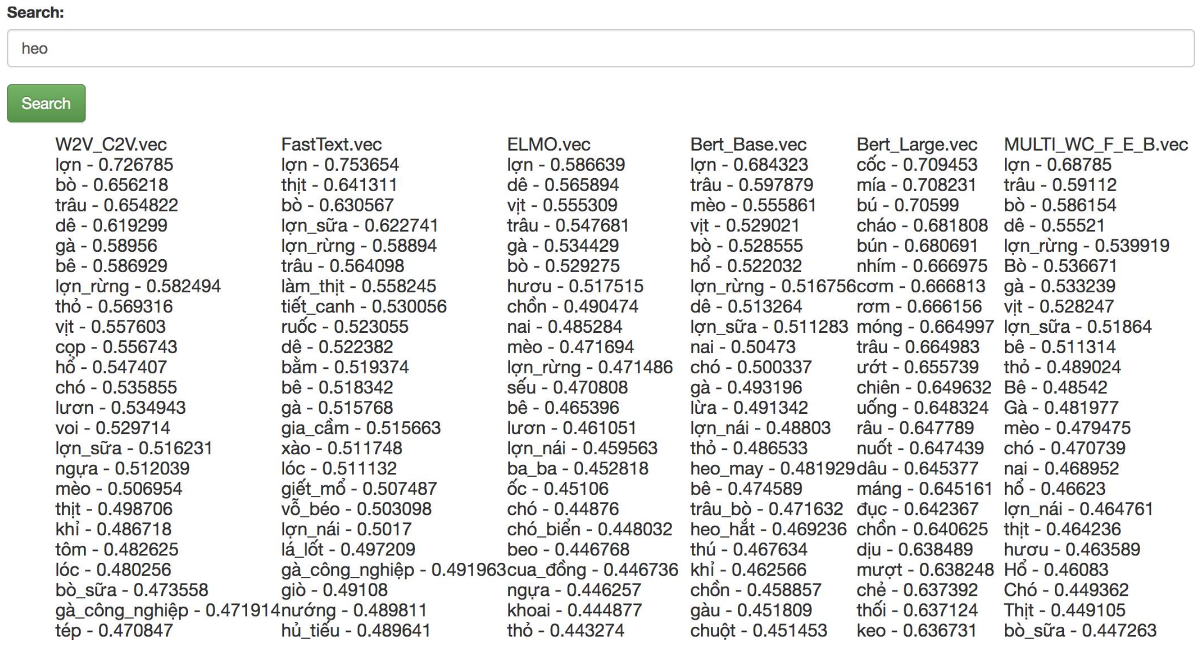
4.1 Training Word Embeddings

We trained embedding models detailed in Section 2 on the Wikipedia dump in Vietnamese[3]. We then apply sentence tokenization and word segmentation provided by VnCoreNLP (Vu et al., 2018a; Nguyen et al., 2018a) to pre-process all documents. It is noted that, for BERT model, we have to (1) format the data differently for the next sentence prediction task; and (2) use SentencePiece (Kudo and Richardson, 2018) to tokenize the data for learning the pre-trained embedding. It is worth

Table 1: Evaluation results of different word embeddings on the Word Analogy Task. P-value column shows significance test results using Paired *t*-tests. '*' means significant (p-value < 0.05) to the rest.

Model	MAP@10	P-value
W2V_C2V	0.4796	*
FastText	0.4970	See [1] & [2]
ELMO	**0.4999**	vs. FastText: 0.95 [1]
BERT_Base	0.4609	*
BERT_Large	0.4634	-
MULTI	0.4906	vs. FastText: 0.025 [2]

[3] https://goo.gl/8WNfyZ



W2V_C2V.vec	FastText.vec	ELMO.vec	Bert_Base.vec	Bert_Large.vec	MULTI_WC_F_E_B.vec
lợn - 0.726785	lợn - 0.753654	lợn - 0.586639	lợn - 0.684323	cốc - 0.709453	lợn - 0.68785
bò - 0.656218	thịt - 0.641311	dê - 0.565894	trâu - 0.597879	mía - 0.708231	trâu - 0.59112
trâu - 0.654822	bò - 0.630567	vịt - 0.555309	mèo - 0.555861	bú - 0.70599	bò - 0.586154
dê - 0.619299	lợn_sữa - 0.622741	trâu - 0.547681	vịt - 0.529021	cháo - 0.681808	dê - 0.55521
gà - 0.58956	lợn_rừng - 0.58894	gà - 0.534429	bò - 0.528555	bún - 0.680691	lợn_rừng - 0.539919
bê - 0.586929	trâu - 0.564098	bò - 0.529275	hổ - 0.522032	nhím - 0.666975	Bò - 0.536671
lợn_rừng - 0.582494	làm_thịt - 0.558245	hươu - 0.517515	lợn_rừng - 0.516756	cơm - 0.666813	gà - 0.533239
thỏ - 0.569316	tiết_canh - 0.530056	chồn - 0.490474	dê - 0.513264	rơm - 0.666156	vịt - 0.528247
vịt - 0.557603	ruốc - 0.523055	nai - 0.485284	lợn_sữa - 0.511283	móng - 0.664997	lợn_sữa - 0.51864
cọp - 0.556743	dê - 0.522382	mèo - 0.471694	nai - 0.50473	trâu - 0.664983	bê - 0.511314
hổ - 0.547407	bằm - 0.519374	lợn_rừng - 0.471486	chó - 0.500337	ướt - 0.655739	thỏ - 0.489024
chó - 0.535855	bê - 0.518342	sếu - 0.470808	gà - 0.493196	chiên - 0.649632	Bê - 0.48542
lươn - 0.534943	gà - 0.515768	bê - 0.465396	lừa - 0.491342	uống - 0.648324	Gà - 0.481977
voi - 0.529714	gia_cầm - 0.515663	lươn - 0.461051	lợn_nái - 0.48803	râu - 0.647789	mèo - 0.479475
lợn_sữa - 0.516231	xào - 0.511748	lợn_nái - 0.459563	thỏ - 0.486533	nuốt - 0.647439	chó - 0.470739
ngựa - 0.512039	lóc - 0.511132	ba_ba - 0.452818	heo_may - 0.481929	dâu - 0.645377	nai - 0.468952
mèo - 0.506954	giết_mổ - 0.507487	ốc - 0.45106	bê - 0.474589	máng - 0.645161	hổ - 0.46623
thịt - 0.498706	vỗ_béo - 0.503098	chó - 0.44876	trâu_bò - 0.471632	đục - 0.642367	lợn_nái - 0.464761
khỉ - 0.486718	lợn_nái - 0.5017	chó_biển - 0.448032	heo_hắt - 0.469236	chồn - 0.640625	thịt - 0.464236
tôm - 0.482625	lá_lốt - 0.497209	beo - 0.446768	thú - 0.467634	dịu - 0.638489	hươu - 0.463589
lóc - 0.480256	gà_công_nghiệp - 0.491963	cua_đồng - 0.446736	khỉ - 0.462566	mượt - 0.638248	Hổ - 0.46083
bò_sữa - 0.473558	giò - 0.49108	ngựa - 0.446257	chồn - 0.458857	chẻ - 0.637392	Chó - 0.449362
gà_công_nghiệp - 0.471914	nướng - 0.489811	khoai - 0.444877	gàu - 0.451809	thối - 0.637124	Thịt - 0.449105
tép - 0.470847	hủ_tiếu - 0.489641	thỏ - 0.443274	chuột - 0.451453	keo - 0.636731	bò_sữa - 0.447263

Figure 5: Side-by-side visualization for the word “heo $_{\textbf{pig}}$” with multiple embeddings. From this visualization, we get the sense that W2V_C2V, ELMO, and Bert_Base mainly capture the categorical information (i.e., “heo $_{\textbf{pig}}$” is surrounded by names of other animals, e.g., "bò $_{\textbf{cow}}$", "trâu $_{\textbf{buffalo}}$") while “FastText“ captures both categorical information (i.e., surrounded by names of other animals) and related verbs to “pig” such as “xào $_{\textbf{frying}}$”, “nướng $_{\textbf{grill}}$”. Bert_Large, on the other hand, does not converge well due to the short training steps mentioned in section 4, therefore, many irrelevant words (e.g., “cốc $_{\textbf{cup}}$”, “dịu $_{\textbf{floppy}}$”) are surrounded the input word “heo $_{\textbf{pig}}$”, “keo$_{\textbf{glue}}$”.

Table 2: Example of five types of semantic and four (out of nine) types of syntactic questions in the word analogy list. “NOT AVAILABLE“ means that the syntactic phenomena do not apply in Vietnamese in comparison to the list of Mikolov et al. (2013).

<table>
<tr><th></th><th>Type of relationship</th><th>Word Pair 1</th><th>Word Pair 2</th></tr>
<tr><td>Semantic</td><td>capital-common-countries</td><td>Athens | Hy_Lạp $_{\text{Greek}}$</td><td>| Baghdad | Irac</td></tr>
<tr><td></td><td>capital-world</td><td>Abuja | Nigeria</td><td>| Thổ Nhĩ Kỳ $_{\text{Turkey}}$ | Turkey</td></tr>
<tr><td></td><td>currency</td><td>Algeria | dinar</td><td>| Canada | đô la $_{\text{dollar}}$</td></tr>
<tr><td></td><td>city-in-zone</td><td>Hòa Bình $_{\text{Hoa Binh}}$ | Tây Bắc Bộ $_{\text{West North}}$</td><td>| Hà Giang $_{\text{Ha Giang}}$ | Đông Bắc Bộ $_{\text{East Northern}}$</td></tr>
<tr><td></td><td>family</td><td>cậu bé $_{\text{boy}}$ | cô gái $_{\text{girl}}$</td><td>| anh trai $_{\text{brother}}$ | em gái $_{\text{sister}}$</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td>Syntactic</td><td>gram1-adjective-to-adverb</td><td colspan="2">NOT AVAILABLE</td></tr>
<tr><td></td><td>gram2-opposite</td><td>chấp nhận được $_{\text{acceptable}}$ | không thể chấp nhận $_{\text{unacceptable}}$</td><td>| nhận thức $_{\text{aware}}$ | không biết $_{\text{unaware}}$</td></tr>
<tr><td></td><td>gram3-comparative</td><td>tệ $_{\text{bad}}$ | tệ hơn $_{\text{worse}}$</td><td>| lớn $_{\text{big}}$ | lớn hơn $_{\text{bigger}}$</td></tr>
<tr><td></td><td>gram4-superlative</td><td>lớn $_{\text{big}}$ | lớn nhất $_{\text{biggest}}$</td><td>| sáng $_{\text{bright}}$ | sáng nhất $_{\text{brightest}}$</td></tr>
<tr><td></td><td>gram5-present-participle</td><td colspan="2">NOT AVAILABLE</td></tr>
<tr><td></td><td>gram6-nationality-adjective</td><td>Albania | Tiếng Albania $_{\text{Albanian}}$</td><td>| Argentina | Tiếng Argentina $_{\text{Argentinean}}$</td></tr>
<tr><td></td><td>gram7-past-tense</td><td colspan="2">NOT AVAILABLE</td></tr>
<tr><td></td><td>gram8-plural-nouns</td><td colspan="2">NOT AVAILABLE</td></tr>
<tr><td></td><td>gram9-plural-verbs</td><td colspan="2">NOT AVAILABLE</td></tr>
</table>

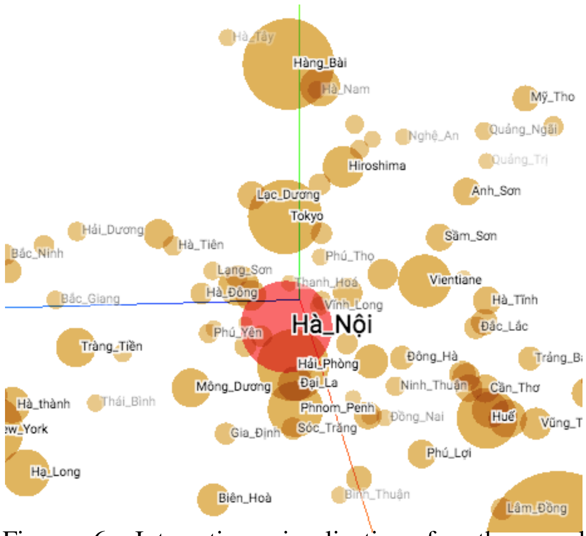


Figure 6: Interactive visualization for the word "Hà_Nội" with ELMO embeddings where near "Hà_Nội" are the names of many other cities in Vietnam (e.g., "Hải_Phòng **Hai Phong**" as well as capital of other countries (e.g., Tokyo).

Table 3: Grid search for hyper-parameters.

Hyper-parameter	Search Space		
cemb dim (char embedding)	50	100	500
drpt (dropout rate)	0.3	0.5	0.7
lstm-s (LSTM size)	50	100	500
lrate (learning rate)	0.0005	0.001	0.005

noting that due to the limitation in computing resources, we can only run BERT_Base for 900,000 update steps and BERT_Large for 60,000 update steps. We, therefore, do not report the result of BERT_Large for a fair comparison. We also create *MULTI* embeddings by concatenating four sets of embeddings (i.e., W2V_C2V, fastText, ELMO and BERT_Base) [4].

4.2 Dataset

The named entity recognition (NER) shared task at the 2016 VLSP workshop provides a dataset of 16,861 manually annotated sentences for training and development, and a set of 2,831 manually annotated sentences for test, with four NER labels PER, LOC, ORG, and MISC. The data was published in 2016 and recently reported in Nguyen et al. (2019). It is a standard benchmark on the NER task and has been used in (Vu et al., 2018a; Dong and Nguyen, 2018). It is noted that, in the original dataset, each word representing a full personal name are separated into syllables that constitute the word. Because this annotation scheme results in an unrealistic scenario for a pipeline evaluation (Vu et al., 2018a), therefore, we tested on a "modified" VLSP 2016 corpus where we merge contiguous syllables constituting a full name to form a word. This similar setup was also used in (Vu et al., 2018a; Dong and Nguyen, 2018), the current state-of-the-art approaches.

4.3 Word Analogy Task

To measure the quality of different sets of embeddings in Vietnamese, similar to Mikolov et al. (2013), we define a word analogy list consisting of 9,802 word analogy records. To create the list, we selected suitable categories from the English word analogy list and then translated them to Vietnamese. We also added customized categories which are suitable for Vietnamese (e.g., cities and their zones in Vietnam). Different from (Mikolov et al., 2013), five categories: "Adjective to adverb", "Present Participle", "Past tense", "Plural nouns", "Plural verbs" were not used to be translated in Vietnamese since the same syntactic phenomena does not exist in Vietnamese. Table 2 shows the list of categories and their examples of the constructed word analogy list in Vietnamese. Since most of this process is automatically done, it can be applied easily to other languages. To know which set of word embeddings potentially works better for a target downstream task, we limit the vocabulary of the embeddings similar to vocabulary of the task (i.e., the NER task). Thus, only 3,135 word analogy records are being evaluated for the NER dataset (Section 4.2).

Regarding the evaluation metric, Mikolov et al. (2013) used accuracy metric to measure the quality of word embeddings on the task in which only when the expected word is on top of the prediction list, then the model gets +1 for true positive count. However, this is not a well-suited metric in low resource languages where training corpus is relatively small, i.e., 233M tokens in Vietnamese Wiki compared to 6B tokens in Google News corpus. Therefore, we change to use mean average precision (MAP) metric to measure quality of the word analogy task. MAP is widely used in information retrieval to evaluate results based on the top$K$ returned results (Vu et al., 2019). We use MAP@10 in this paper. Table 1 shows evaluation results of different sets of embeddings on the word analogy task. The *evaluator* of ETNLP also shows P-value using the paired t-tests on the raw

[4] We do not use W2V here because W2V_C2V is W2V with the use of character embedding to deal with OOV.

Table 4: Performance of the NER task using different embedding models. The $MULTI_{WC\_F\_E\_B}$ is the concatenation of four embeddings: W2V_C2V, fastText, ELMO, and Bert_Base. "wemb dim" is the dimension of the embedding model. VnCoreNLP* means we retrain the VnCoreNLP with our pre-trained embeddings.

	F1	wemb dim	cemb dim	drpt	lstm-s	lrate
BiLC3 (Ma and Hovy, 2016)	88.28	300	-	-	-	-
VNER (Dong and Nguyen, 2018)	89.58	300	300	0.6	-	0.001
VnCoreNLP (Vu et al., 2018a)	88.55	300	-	-	-	-
VnCoreNLP $^{(*)}$	**91.30**	1024	-	-	-	-
BiLC3 + W2V	*89.01*	300	50	0.5	100	0.0005
BiLC3 + BERT-Base	*88.26*	768	500	0.3	100	0.0005
BiLC3 + W2V_C2V	*89.46*	300	100	0.5	500	0.0005
BiLC3 + fastText	89.65	300	500	0.3	100	0.001
BiLC3 + ELMO	89.67	1024	100	0.7	500	0.0005
BiLC3 + $\mathrm{MULTI}_{WC\_F\_E\_B}$	**91.09**	2392	100	0.7	100	0.001

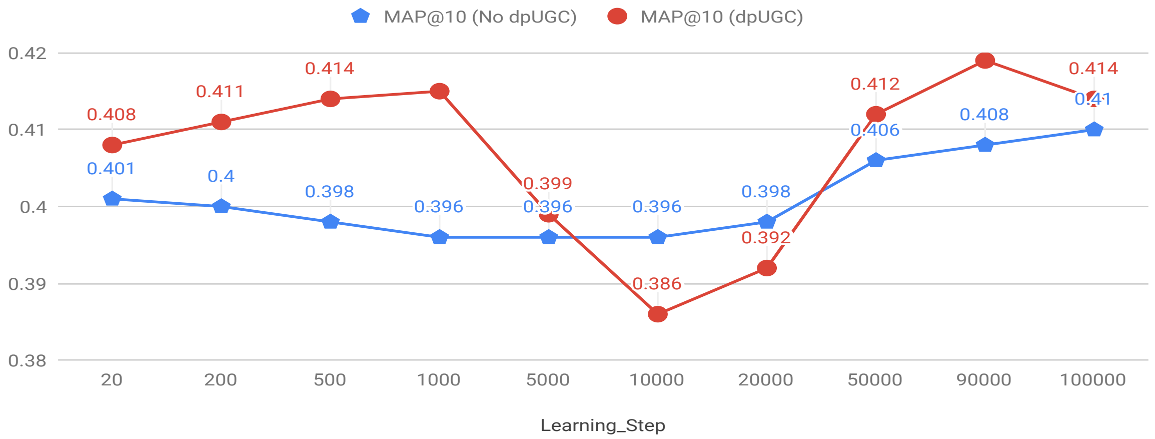


Figure 7: Evaluation results of different word embeddings trained using **dpUGC** and **No dpUGC** (i.e., one option in dpUGC to train embeddings without privacy guarantee for comparison) on the Word Analogy Task.

MAP@10 scores (i.e., before averaging) between different sets of embeddings. The P-values (Table 1) show that the performances of the top three sets of word embeddings (i.e., fastText, ELMO, and MULTI), are significantly better than the remainders but there is no significant difference between the three. Therefore, these sets of embeddings will be selected for NER task.

4.4 NER Task

Model: We apply the current most well-known neural network architecture for NER task of Ma and Hovy (2016) with no modification in its architecture, namely, *BiLSTM-CRF+CNN-char (BiLC3)*. Only in the embedding layer, a different set of word embeddings is used to evaluate their effectiveness. Regarding experiments, we perform a grid search for hyper-parameters and select the best parameters on the validation set to run on the test set. Table 3 presents the value ranges we used to search for the best hyper-parameters. We also follow the same setting as in (Vu et al., 2018a) to use the *last* 2000 records in the training data as the validation set. Moreover, due to the availability of the VnCoreNLP code, we also retrain their model with our pre-trained embeddings (*VnCoreNLP**).

Main Results: Table 4 shows the results of NER task using different word embeddings. It clearly shows that, by using the pre-trained embeddings on Vietnamese Wikipedia data, we can achieve the new state-of-the-art results on the task. The reason might be that FastText, ELMO and MULTI can handle OOV words as well as capture better the context of the words. Moreover, learning the embeddings from a formal dataset like Wikipedia is beneficial for the NER task. This also verified the fact that using our pre-trained embeddings on VnCoreNLP helps significantly boost its performance. Table 4 also shows the F1 scores of W2V, W2V_C2V and BERT_Base embeddings which are worse than three *selected* embeddings

Table 5: P-values of the paired t-tests between embeddings obtained using dpUGC at different learning step (Emb@L). "-" denotes values of these entries in the upper triangular matrix are the values of the transposed entries in the lower triangular matrix. P-values in **bold font** are statistical significance at the level of 0.05.

Emb@L	20	200	500	1000	5000	10000	20000	50000	90K	100K
20	1	-	-	-	-	-	-	-	-	-
200	0.0578	1	-	-	-	-	-	-	-	-
500	**0.0074**	0.1809	1	-	-	-	-	-	-	-
1000	**0.0053**	0.169	0.9031	1	-	-	-	-	-	-
5000	**0.0178**	**0.0009**	6.992	1.6242	1	-	-	-	-	-
10000	2.543	6.9872	2.25867	9.3987	**0.001**	1	-	-	-	-
20000	**0.0016**	**0.0001**	1.757	9.6053	0.112	0.1819	1	-	-	-
50000	0.5077	0.9023	0.73137	0.7003	**0.031**	5.0673	**0.0001**	1	-	-
90K	0.1205	0.2878	0.5127	0.5323	**0.0049**	2.4211	**0.0001**	0.2688	1	-
100K	0.3777	0.6822	0.9932	0.9764	**0.0357**	8.2638	**0.0019**	0.7274	0.2758	1

(i.e., fastText, ELMO and MULTI). This might indicate that using word analogy to select embeddings for downstream NLP tasks is sensible.

4.5 Privacy-Guaranteed Embedding Selection Task

In this section, we show how to apply ETNLP to another downstream task of privacy-guaranteed embedding selection. Vu et al. (2019) introduced dpUGC to guarantee privacy for word embeddings. The main intuition behind dpUGC is that, when the embedding is trained on very sensitive text corpus (e.g., medical text data), it has to guarantee privacy at the highest level to prevent privacy leakage. However, among many embeddings at different learning steps of dpUGC, how to choose a suitable embedding to achieve a good trade-off between data privacy and data utility is a key challenge. To this end, we propose to apply ETNLP into this scenario to select good embeddings for knowledge sharing using dpUGC.

Similar to Vu et al. (2019), we trained 20 different embeddings from 10 different learning steps while training on the same Vietnamese Wikipedia dataset as used in Section 4.1 with (dpUGC) and without privacy-guarantee (No dpUGC) to evaluate their performances. Figure 7 shows that the pre-trained embedding at learning_step 1000 (Emb@1000) seems to be a good word embedding candidate to have a good trade-off between privacy guarantee and data utility. Emb@1000 was in favor because of two reasons. Firstly, in training privacy-guaranteed embeddings, we try to stop as early as possible since the more training steps we run, the higher privacy we have to sacrifice (Vu et al., 2019). Secondly, its performance in the Word Analogy Task was more or less similar to the other good embedding at the learning step 90K (i.e., Emb@90K). In fact, from Table 5 we know that the performance between Emb@1000 and Emb@90K learning steps are not significant difference. Therefore, selecting the pre-trained embedding at the learning step 1000 is the best option for privacy-guaranteed embedding using dpUGC. In summary, in this task, we showed how ETNLP can be used to select a good word embedding candidate for privacy-guaranteed knowledge sharing. Normally, this selection process is very time consuming, however, it is much easier with ETNLP since it allows users to import multiple embeddings for running evaluations.

5 Conclusions

We have presented a new systematic pipeline, ETNLP, for extracting, evaluating and visualizing multiple pre-trained embeddings on a specific downstream task. The ETNLP pipeline was designed with three principles in mind: (1) easy to apply on any language processing task, (2) better performance, and (3) be able to handle unknown vocabulary in real-world data (i.e., using C2V (char to vec)). The evaluation of the approach in (1) Vietnamese NER task and (2) privacy-guaranteed embedding selection task showed its effectiveness.

In the future, we plan to support more embeddings in different languages, especially in low resource languages. We will also support new ways to explore the embedding spaces including at phrase and subword levels.

Acknowledgement

This work is supported by the Federated Database project funded by Umeå University, Sweden. The computations were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at HPC2N.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 809–815. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

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

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

Ngan Dong and Kim Anh Nguyen. 2018. Attentive neural network for named entity recognition in vietnamese. *CoRR*, abs/1810.13097.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. Character-aware neural language models. *CoRR*, abs/1508.06615.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *CoRR*, abs/1808.06226.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2177–2185, Cambridge, MA, USA. MIT Press.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Dat Quoc Nguyen, Richard Billingsley, Lan Du, and Mark Johnson. 2015. Improving Topic Models with Latent Feature Word Representations. *Transactions of the Association for Computational Linguistics*, 3:299–313.

Dat Quoc Nguyen, Dai Quoc Nguyen, Thanh Vu, Mark Dras, and Mark Johnson. 2018a. A Fast and Accurate Vietnamese Word Segmenter. In *Proceedings of the 2018 LREC*, Miyazaki, Japan.

Huyen Nguyen, Quyen Ngo, Luong Vu, Vu Tran, and Hien Nguyen. 2019. Vlsp shared task: Named entity recognition. *Journal of Computer Science and Cybernetics*, 34(4):283–294.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2018b. Introducing two vietnamese datasets for evaluating semantic models of (dis-)similarity and relatedness. In *Proceedings of the 2018 NAACL: Short Papers*, pages 199–205.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 338–348, Copenhagen, Denmark.

Matt Taddy. 2015. Document classification by inversion of distributed language representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 45–49. Association for Computational Linguistics.

Thanh Vu, Dat Quoc Nguyen, Dai Quoc Nguyen, Mark Dras, and Mark Johnson. 2018a. Vncorenlp: A vietnamese natural language processing toolkit. In *Proceedings of the 2018 NAACL: Demonstrations*, pages 56–60, New Orleans, Louisiana. Association for Computational Linguistics.

Thanh Vu, Dat Quoc Nguyen, Xuan-Son Vu, Dai Quoc Nguyen, Michael Catt, and Michael Trenell. 2018b. Nihrio at semeval-2018 task 3: A simple and accurate neural network model for irony detection in twitter. In *Proceedings of SemEval 2018*.

Xuan-Son Vu, Hyun-Je Song, and Seong-Bae Park. 2014. Building a vietnamese sentiwordnet using vietnamese electronic dictionary and string kernel. In *Knowledge Management and Acquisition for Smart Systems and Services*, pages 223–235, Cham. Springer International Publishing.

Xuan-Son Vu, Son N. Tran, and Lili Jiang. 2019. dpugc: Learn differentially private representation for user generated contents. In *Proceedings of the 20th International Conference on Computational Linguistics and Intelligent Text Processing*.