Analysis on Filipino Word Embeddings

CSC714M - Theories in Natural Language Processing

Paolo Edni Andryn V. Espiritu

De La Salle University

1 Abstract

The development of word embeddings throughout the years have advanced various research areas in natural language processing (NLP). These are representations of a word where its semantic meanings are also captured in the form of a vector in multi-dimensional space. The distance of a vector from another indicates the similarity between the two word representations.

In this study, an analysis between two word embedding techniques will be explored — Word2Vec and FastText. The models were downloaded from Dan John Velasco's GitHub repository. These will be evaluated on the scores it achieved from predicting the top-10 most similar word similarities or analogies. Overall, the results show that Word2Vec provided better predictions compared to the FastText model.

2 Description of the Models

2.1 Word2Vec

Word2Vec is a word embedding technique that was developed and published by Google in 2013. It represents each distinct word in the corpus as a vector of numbers in a multi-dimensional space. Each vector contains the word's semantic meaning based on a span of n-words surrounding it. To measure the similarity between words, cosine similarity is used which implies that the two vectors are nearby one another in the vector space. Its architecture includes the implementation of Continuous Bag of Words (CBoW) and Skip-gram. To briefly describe the two methods, CBoW is aims to reduce the distance between the target and actual words by predicting a target word based on its context. Conversely, skip-gram is used to predict the context words given the target word and its goal is to minimize the difference between the actual and target context words.

2.2 FastText

On the other hand, FastText is an open-sourrce library created by Facebook's AI Research (FAIR) Lab that may also be used for training word embeddings and performing text classification tasks. In this paper, the term 'FastText' will be used to refer to the word embedding model which extends from the features of a Word2Vec model. In contrast to Word2Vec, FastText considers each word as a bag of character n-gram. For instance, the word *'hello'* at $n = 3$ will be broken down into {hel, ell, llo}.

3 Experimental Set-up

The word embedding models used in this assignment were extracted from Dan John Velasco's GitHub repository.

3.1 Word Embedding Corpus

The author noted that the corpora used for training the models were the following [1]:

- WikiText-TL-39
- NewsPH-NLI
- Unpublished Twitter dataset

Before using these datasets, the author also performed the following pre-processing operations:

1. Retain stop words
2. Retain commas
3. Lowercase text
4. Removed quotes and its contents
5. Removed brackets, parenthesis, braces, and its contents
6. Removed punctuations
7. Removed symbols
8. Replace numbers with *xx_digit*
9. Replace amounts with *xx_amount*
10. Replace percentages with *xx_percentage*

As a result, the following are the statistics of the corpora after pre-processing:

- 4.68 million sentences
- 14.28 average sentence length
- 66.9 million tokens
- 1.08 million unique tokens

3.2 Models Used

The models were loaded using the *gensim* library as shown in the GitHub repository [1]. The following are the two models used in this study along with the dimensions, vocabulary size, and file size:

Models	Dimensions	Vocabulary Size	File Size
Word2Vec	300	126,687	269.9 MB
FastText	300	126,687	2.34 GB

Table 1: Word2Vec and FastText models

3.3 Evaluation

Two sets of test cases were prepared. The first set will gauge the performance of the model in predicting the top-10 most similar words based on the input. Whereas, the second set will test the capabilities of the model in providing the missing word for the given analogy. In Table 3, the types of analogies are synonymy, antonymy, part-whole, superclass, and geography from numbers 1 to 5 respectively.

To evaluate the models, the function *most_similar* was used under the *gensim* library. For word similarity, only two arguments were supplemented to the function, *positive* and *topn*, where *positive* contributes positively towards similarity and *topn* shows the resulting top-n words. For word analogies, another argument named *negative* was also used to specify the word that negatively affects the similarity of the first word. For example, given test 2 in Table 3, *buhay* and *laki* will be passed on to the function as *positive* then *patay* will be passed as *negative* to indicate that *laki* should correspond to *liit*

Test	Sample word
1	*gamot*
2	*ilaw*
3	*tubig*
4	*tao*
5	*pusa*

Table 2: List of word test cases

4 Analysis of Results

The tables discussed in this section contain outputs that were ranked from 1 to 10 where 1 is the best result of the model on the given task. For 4.1, the headers of the table indicate the input to the model. Furthermore, for 4.2, the headers of the table indicate the analogy given to the model and the blank line will be predicted by the model.

Test	**Sample analogy**	**Expected Output**
1	*Pinggan is to plato, as gwapo is to* ______	*pogi*
2	*Buhay is to patay, as laki is to* ______	*liit*
3	*Papel is to libro, as mata is to* ______	*mukha*
4	*Talong is to gulay, as asul is to* ______	*kulay*
5	*Australia is to Canberra, as Thailand is to* ______	*bangkok*

Table 3: List of analogy test cases

4.1 Word Similarity

Tables 4 and 5 show the results of Word2Vec and FastText models on predicting the most similar word given an input. As shown in the tables, Word2Vec produced better outputs in terms of semantic similarity to the input. This shows its effectiveness in understanding semantic relationships by using word-level vectors. On the other hand, the FastText model generated similar words based on the sequence of the word. For instance, the results under the *ilaw* column in Table 5 would always contain the substring *ilaw* in each of its output. It can be observed that it does not prioritize the semantic meaning of the input when predicting the most similar word.

One key difference of Word2Vec and FastText is that the FastText model uses subword-level vectors. As previously mentioned, this implies that FastText generate word embeddings by considering character n-grams in each word. With this, the model is often used to better represent embeddings in morphologically-rich languages such as Filipino. This is evident in the column of *gamot* where various forms of the word were also predicted by the model (e.g. *panggamot*, *manggamot*, *pagamot*, *gagamot*).

Rank	***gamot***	***ilaw***	***tubig***	***tao***	***pusa***
1	antibiotic (0.6556)	*kuryente* (0.5558)	*kuryente* (0.6309)	*taong* (0.7032)	*aso* (0.8511)
2	antibiotics (0.6533)	*kandila* (0.5447)	*hangin* (0.6051)	*mamamayan* (0.5896)	*kuting* (0.6781)
3	*alak* (0.6217)	aircon (0.5371)	*gripo* (0.5996)	*babae* (0.5783)	*babae* (0.6219)
4	*pagkain* (0.6126)	electricfan (0.5120)	*yelo* (0.5942)	*bagay* (0.5742)	*langgam* (0.5901)
5	*inumin* (0.6039)	*liwanag* (0.5055)	*kumukulong* (0.5640)	*lalake* (0.5586)	hamster (0.6309)
6	*gatas* (0.5966)	*binata* (0.5049)	*tubig-baha* (0.5634)	*pilipino* (0.5495)	*langaw* (0.5482)
7	vitamins (0.5947)	*kable* (0.5007)	*maiinom* (0.5615)	*bata* (0.5455)	*bubuyog* (0.5479)
8	biogesic (0.5868)	*kawad* (0.4946)	*dugo* (0.5602)	*nilalang* (0.5427)	*lamok* (0.5461)
9	*bakuna* (0.5765)	*kurtina* (0.4879)	*inuming* (0.5585)	*lalaki* (0.5294)	*ipis* (0.5432)
10	*antibiyotiko* (0.5672)	*tubig* (0.4759)	*pagkain* (0.5567)	*indibidwal* (0.4961)	*kambing* (0.5425)

Table 4: Word2Vec word similarity results

4.2 Word Analogy

In tables 6 and 7, the models results for completing the analogy are illustrated. From left to right excluding the rank column, the type of analogy used in each column is synonymy, antonymy, part-whole, superclass, and geography. The target words for these types of analogies are gwapo, liit, mukha, kulay, and bangkok respectively.

Consistent to the findings discussed in 4.2, the Word2Vec model completed the analogy by providing its best prediction of a word that is semantically similar to the expected output. For table 6, it can be observed that the model performed well on synonymy and antonymy since the model produced the expected words which are *pogi* (synonymous with *gwapo*) and *liit* (antonym of *laki*).

Rank	***gamot***	***ilaw***	***tubig***	***tao***	***pusa***
1	*gamotea* (0.8889)	*pailaw* (0.7953)	*tubig** (0.9671)	*taoo* (0.8426)	*aso* (0.8613)
2	*panggamot* (0.8157)	*ilawom* (0.7897)	*tubigg* (0.9416)	*taod** (0.8217)	*pusakal* (0.7980)
3	*pampagamot* (07876)	*pilaw* (0.7763)	*tubig-dagat* (0.8721)	*taoos* (0.8001)	*pusan* (0.7869)
4	*manggamot* (0.7831)	*tilaw* (0.7545)	*catubig* (0.8658)	*taooo* (0.7624)	*pusaaa* (0.7694)
5	*paggamot* (0.7692)	*iilaw* (0.7525)	*patubig* (0.8652)	*taong* (0.7246)	*pusanggala* (0.7630)
6	*pagamot* (0.7635)	*ilawa* (0.7512)	*tubig-baha* (0.8589)	*taoooo* (0.7239)	*daga* (0.7319)
7	*gamos* (0.7585)	*madilaw* (0.7464)	*tubig-alat* (0.8560)	*taob* (0.7122)	*pusang* (0.7298)
8	*gagamot* (0.7569)	*kilaw* (0.7270)	*matubig* (0.8499)	*taoyuan* (0.7103)	*pusaaaa* (0.7295)
9	*gamo* (0.7512)	*umiilaw* (0.7231)	*tubi* (0.8461)	*tao'y* (0.7097)	*kambing* (0.7218)
10	*panggagamot* (0.7384)	*ilawn* (0.7193)	*tubigan* (0.8407)	*taong-bayan* (0.6994)	*manol* (0.7201)

Table 5: FastText word similarity results

For table 7, the results of the FastText model on word analogies. As anticipated, the FastText model performed poorly on word analogies since it only based on the similarity of word sequences. It may also be noticed that FastText returned outputs with typographical errors. Despite producing erroneous outputs, this occurrence showcases one of the strengths of FastText since it can generate embeddings for words that did not exist during its training phase. This attribute is formally referred to as subword information where linguistic units smaller than words may be used to generate meaningful word embeddings.

5 Conclusion

In conclusion, this paper shows the differences between two word embedding techniques — specifically, Word2Vec and FastText. Through the experiments, it was observed that Word2Vec provided better predictions when it comes to semantic relationships of words. With the use of word-level vectors, it showed how words are similar to one another based on the context of its surrounding words.

Conversely, the FastText model predicted the most similar words based on a character-level. It was also evident that the model often predicted typographical errors and most of it are words that are not present in its training data. This shows that FastText can handle out-of-vocabulary words and better represent languages that are morphologically-rich.

References

[1] Velasco, D. (2021). Filipino Word Embeddings. *GitHub*. Retrieved from https://github.com/danjohnvelasco/Filipino-Word-Embeddings.

Rank	**p**inggan : plato | gwapo : ___	**b**uhay : patay | laki : ___	**p**apel : libro | mata : ___	**t**along : gulay | asul : ___	**a**ustralia : canberra | thailand : ___
1	*pogi* (0.7468)	*liit* (0.4497)	*paningin* (0.4956)	*kahel* (0.4755)	japan (0.5281)
2	*ampogi* (0.5761)	*anlaki* (0.4467)	*braso* (0.4918)	*berde* (0.4515)	india (0.5260)
3	*gwapooo* (0.5610)	*napakalaki* (0.4350)	*pisngi* (0.4549)	*pula* (0.4441)	vietnam (0.5219)
4	*napakagwapo* (0.5427)	*bohai* (0.4337)	*ilong* (0.4541)	*abuhing* (0.4284)	indonesia (0.5177)
5	*cute* (0.5410)	*malaki* (0.4160)	*leeg* (0.4502)	*matingkad* (0.4224)	taiwan (0.5177)
6	*gwapoo* (0.5349)	*lahat* (0.3835)	*dibdib* (0.4458)	*puting* (0.4200)	singapore (0.5071)
7	*popogi* (0.5269)	*laking* (0.3755)	*tainga* (0.4251)	*pulang* (0.4197)	malaysia (0.5032)
8	*gwapoooo* (0.5193)	*buhai* (0.3697)	*kamay* (0.4223)	*itim* (0.4195)	myanmar (0.4905)
9	*pogiii* (0.5109)	*napakaliit* (0.3693)	*matang* (0.4221)	*berdeng* (0.4162)	turkey (0.4795)
10	*gwapooooo* (0.5017)	*paglaki* (0.3656)	*ulo* (0.4174)	*matitingkad* (0.4131)	korea (0.4692)

Table 6: Word2Vec analogy results

Rank	**p**inggan : plato | gwapo : ___	**b**uhay : patay | laki : ___	**p**apel : libro | mata : ___	**t**along : gulay | asul : ___	**a**ustralia : canberra | thailand : ___
1	*gwapo-gwapo* (0.8186)	*laki-laki* (0.6621)	*paningin* (0.6699)	*rasul* (0.5702)	australians (0.6875)
2	*g-gwapo* (0.7936)	*mlaki* (0.6124)	*mapapel* (0.6687)	*sul* (0.5470)	australis (0.6754)
3	*gugwapo* (0.7936)	*anlaki* (0.5820)	*takipmata* (0.6683)	*puting* (0.54541)	australasia (0.6733)
4	*pogi* (0.7768)	*laking* (0.5781)	*pilikmata* (0.6572)	*polong* (0.5428)	japan (0.6641)
5	*gwagwapo* (0.7765)	*lakin* (0.5688)	*namamalikmata* (0.6522)	*itim* (0.5382)	singapore (0.6621)
6	*ga-gwapo* (0.7635)	*kalaki* (0.5620)	*pagkatawan* (0.6471)	*berdeng* (0.5318)	indonesia (0.65971)
7	*ggwapo* (0.7601)	*anlaking* (0.5489)	*matamlay* (0.6470)	*talolong* (0.5256)	australian (0.6457)
8	*gagwapo* (0.7546)	*lakim* (0.5488)	*kamatyan* (0.6458)	*talon-talon* (0.5174)	malaysia (0.6355)
9	*gwapoo* (0.7515)	*buhay* (0.5481)	*pamata* (0.6440)	*hasul* (0.5172)	singaporeans (0.6304)
10	*angwapo* (0.7433)	*apakalaki* (0.5443)	*pakatawa* (0.6436)	*zilong* (0.5166)	australoid (0.6274)

Table 7: FastText analogy results