2022 年臺灣國際科學展覽會
優勝作品專輯

作品編號 **190039**

參展科別 電腦科學與資訊工程

作品名稱 **Cross-lingual Information Retrieval**

得獎獎項

國　　家 **Singapore**

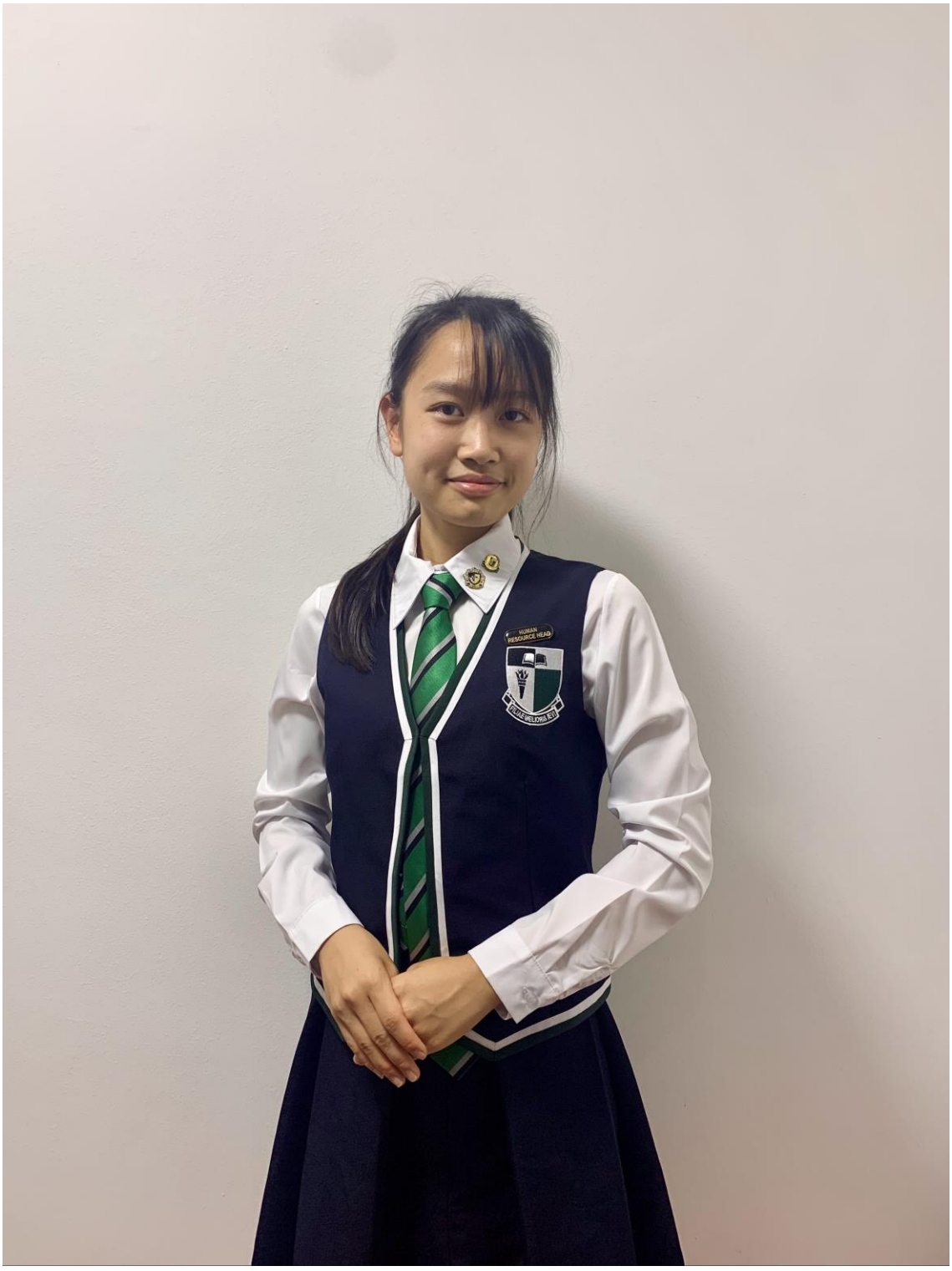
就讀學校 **Raffles Girls' School**

指導教師 **Shaun de Souza**

作者姓名 **Alysa Lee Mynn**

關鍵詞 **Cross Lingual Information Retrieval CLIR)、Cross-lingual word embeddings、random shuffling**

作者照片



***Abstract*—In this project, we evaluate the effectiveness of Random Shuffling in the Cross Lingual Information Retrieval (CLIR) process. We extended the monolingual Word2Vec model to a multilingual one via the random shuffling process. We then evaluate the cross-lingual word embeddings (CLE) in terms of retrieving parallel sentences, whereby the query sentence is in a source language and the parallel sentence is in some targeted language. Our experiments on three language pairs showed that models trained on a randomly shuffled dataset outperforms randomly initialized word embeddings substantially despite its simplicity. We also explored Smart Shuffling, a more sophisticated CLIR technique which makes use of word alignment and bilingual dictionaries to guide the shuffling process. Due to the complexity of the implementation and unavailability of open source codes, we defer experimental comparisons to future work.**

I. INTRODUCTION

In monolingual information retrieval, the queries and answers for retrieval are in the same language. However, in Cross Lingual Information Retrieval (CLIR), queries and answers are in different languages. This increases the difficulty of retrieving answers that are actually of interest to the user. For example, the user may enter a query in English, while the system's goal is to return a ranked list of documents in French that the user is interested in. Existing CLIR approaches include document translation, query translation, as well as the mapping of a both document and query to a third language or medium for comparison. Of the three, query translation is generally considered the most suitable due to its simplicity and effectiveness [1]. However, the main problem is dealing with translation ambiguity, which becomes more pronounced when query sentences are shorter. With limited context, translations to related terms in the document's language would be difficult and inaccurate. In comparison, document translation is computationally more expensive and harder to scale as the entire document has to be translated to query language. However, its allure resides in its increased probability of correct translation to a synonymous query word, especially amongst more common query words. More recently, Vulic and Moens [2] introduced the concept of the Cross-lingual Word Embeddings (CLE) approach, which converts randomly shuffled parallel sentences into word embeddings for comparison. This method proved much more accurate than a model trained with a simpler baseline(translation of query before matching) for several reasons, one of which is due to its ability to efficiently convert parallel texts into dense vectors and map their proximity. The Smart Shuffling method introduced by Hamed, Sheikh and Allen in July 2020[3] takes the CLE approach one step further with the help of a dictionary in the reordering process when shuffling the parallel sentences. In our exploration, we implemented our simplified interpretation of their algorithm and exemplified how it shuffles grouped words with similar definitions closer to each other. In traditional information retrieval, the queries and documents for retrieval are in the same language. However, in Cross Lingual Information Retrieval(CLIR), queries and documents are in different languages. This increases the difficulty of retrieving documents that are actually of interest to the user. Generally, Random Shuffling is used in the CLIR process. Hence we want to explore the effectiveness of this method.

II. FRAMEWORK

A. *Word2Vec Model and its Parameters*

For this experiment, we used Word2Vec, a popular method used to construct word embeddings from words in a document's vocabulary using a shallow neural network. It was developed by Tomas Mikolov in 2013 at Google [4]. The word embedding formed from each word is capable of capturing the context of a word in the document, as well as its semantic and syntactic similarity in relation to the other words. We chose to use the Skip-Gram model. According to Mikolov[5], this model has the ability to represent words well despite working with small amounts of data. Given a target word, the skip gram model tries to predict its context, i.e. the surrounding words. For each input word in the input layer, the input word is linearly transformed through a weight matrix to form its one-hot representation and activated with an activation function to create a hidden layer. Each word also goes through a backward pass(backpropagation) which re-calculates the input and output weight matrices. This process is repeated for every word in the training dataset in order to create word embeddings for use later on in the experiment. For our Word2Vec model, we use hyper-parameters based on the default values. Our minimum count was 3, which meant that in the dataset, a word had to have a total frequency higher than 3 before it would be used for training the model. Our window size was 5 for our skip-gram model, which meant that the maximum distance between the current and predicted word in a sentence would be 5. We set the embedding size to 100.

B. *Random Shuffling*

For the Random Shuffling approach, each parallel sentence pair in the data set was tokenized. The word tokens of each sentence pairs are then randomly shuffled. Thereafter, the shuffled sentence will be used for training Word2Vec. The random shuffling creates a coarse-grained bilingual context for each word and enable the creation of a cross-lingual embedding space. Cross-lingual contexts allows the learned representations to capture cross-lingual relationships. While adjacent words in the shuffled sentences may not be correct translations, or may not approximate the original context closely, we hypothesize that if the sentences are short, random shuffling may still work adequately. Figure 1 illustrates random shuffling. For Word2Vec training, if the contextual window is set high enough, randomly shuffled words can still have a chance of forming useful cross-lingual associations. For example, the word "calling" would form connections with the words around it based on the window size. Due to this, the word "appellant", which is the corresponding French definition

of "calling", would form a strong association with it as well. The random shuffling technique thus may be able to capture cross-lingual information despite its simplicity. The code used can be found in Appendix 1.

French sentence: *Appellant les knights des les d'Orient l'est*

English sentence: *Calling the knights of the Oriental East*

Shuffled Sentence: *de knights les of appellant calling east oriental knights les des the the d'Orient l'est*

Figure 1: Randomly shuffled French-English sentence. In the shuffled sentence, shaded words are from the French sentence, while unshaded words are form the English sentence.

C. *Smart Shuffling*

In comparison, for the Smart Shuffling approach, word tokens are not shuffled randomly. Given a sentence in the source language, words from the parallel sentence in the targeted language are inserted as guided by the following procedure:

- Words with similar forms in both the source and target language are placed adjacent to each other in the shuffled sentence. For example in Figure 2, the word "knight" is similar in both French and English.
- Looking up a cross-lingual dictionary which maps words in a source language to the words in the target language. If there are matches, the target word is inserted adjacent to the source word. For example in Figure 2, the word "Appelant" maps to "Calling".
- If both above cases are not met, computing the character n-gram overlap between the dictionary translation of the source word and the target word, which is then used to compute a probability distribution. Given the source word, the target word is sampled. For example, "d'orient" and "oriental" overlaps substantially. Given "d'orient", "oriental" will be sampled with higher probability for placement beside "d'orient", as compared to other words in the English sentence. • If all above cases are not met, the target word is sampled from a uniform distribution, given the source word. Also note that insertion of target words is adjacent to, but randomly before or after the source word.

French sentence: *Appellant les knights des les d'Orient l'est*

English sentence: *Calling the knights of the Oriental East*

Shuffled Sentence: *appellant les the knights knights of des les the d'Orient oriental l'est East*

Figure 2: An example of a smartly shuffled sentence. The source language is French and the target language is English.

We trained a Word2Vec model, utilising concepts such as random shuffling, cosine similarity, taking the mean reciprocal rank and calculating accuracy of test results.

III. FINDINGS

We applied the Random Shuffling approach to datasets of parallel movie subtitles. We downloaded parallel move subtitles from OPUS [6] and pre-processed it to remove punctuation as well as lower all alphabets. This was so as to ensure that the Word2Vec model learns each word independent of the punctuation around it, and does not mix up punctuations with words. Ensuring that all letters are in lowercase would lessen noise when processing and training the model as the model would not classify the capitalised and lowercase word as different words. For example, 'America' and 'america' would be classified as two different words though they are in actuality one word. Hence, the words should be converted to lowercase so as to minimise noise in the training process and prevent incorrect classification. Thereafter, we conducted 5 trials whereby for each trial, we randomly selected 1000 parallel sentence pairs as the test dataset. For each trial, we evaluate the randomly initialized embeddings first on the test data set prior to training. Thereafter, it was trained on randomly shuffled data before being tested with the 1000 parallel sentence pairs to evaluate its accuracy. The aim of conducting multiple trials was to improve the estimate of the mean model performance. Arbitrarily selecting the randomly shuffled data for training also served as a less-biased representative of the overall dataset.

A. *Testing*

For the testing process, we converted the words in the query and target sentences into word embeddings if they were in the vocabulary of the model. Each sentence was represented by a vector from the average of all the word embeddings it contained.

$$similarity(A,B) = \frac{A \cdot B}{||A|| \times ||B||} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Equation 1: Cosine Similarity

Thereafter, we computed the cosine similarity between test vectors and candidate vectors. Equation 1 illustrates cosine similarity between vectors A and B, each of dimension n, and where ||.|| denotes the Euclidean norm of vectors. Cosine similarity ranges from -1 to 1, where -1 means that the results are perfectly dissimilar whereas 1 is perfectly similar. A cosine value of 0 means that the two vectors are perpendicular to each other. For each test vector, the corresponding candidate vectors were ranked in descending order.

$$Mean\ Reciprocal\ Rank(MRR) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank_i}$$

Equation 2: Mean Reciprocal Rank

After the parallel sentence was sorted in order of descending cosine similarity, we then calculated the Mean Reciprocal Rank(MRR) of the correct candidate vector with Equation (2). For the i-th test sentence, the reciprocal rank is 1/ranki where ranki is the rank of the matching parallel sentence. For multiple queries, the reciprocal rank is the mean of the N reciprocal ranks. MRR is high if the correct corresponding candidate vectors are ranked high. We also computed the top-10 accuracy, whereby for each test sentence, we checked if the correct answer sentence was ranked within the top 10. Across

all test sentences, we counted the number of times the correct candidate vector was within the top 10 candidate vectors and divided that by the number of test vectors to get our accuracy score. Before training, we calculated by probability that the parallel answer vector would have a 0.1% chance of being within the top 10 candidate vectors. We then hypothesised that, given our genism model consistently attained roughly 70% cosine-similarity score between the English word "house" and its French, Spanish and German synonyms ("maison", "casa" and "haus") after training, the top-10 accuracy score would be roughly 70%. In order to evaluate the trained model, the model was tested on 1000 parallel test sentences across 3 different language pairs. The Mean Reciprocal Rank and Accuracy were both derived from the averages of 5 trials for each language pair. For each trial, the model was tested twice, once before and once after training. "Pre-training" refers to the model before it was trained on the bilingual training data while "Post-training" refers to the same model after it has been trained.

B. *Results*

		MRR	**Top-10 Accuracy**
English-French	Pre-training	0.0585	0.0838
	Post-training	0.547	0.673
English-German	Pre-training	0.0387	0.0622
	Post-training	0.505	0.627
English-Spanish	Pre-training	0.0673	0.101
	Post-training	0.696	0.817

Table 1: Table of results. Pre-training refers to randomly initialised embeddings used prior to training. Post-training refers to The trained Word2Vec model based on random shuffling.

C. *Equations*

Table 1 represents our experimental results for the Random Shuffle datasets. On a whole, there was a significantly higher Top-10 Accuracy score after training. Using the English-French results as a benchmark, the Top-10 Accuracy score increased from 0.0838 to 0.673 after training. This means that after training, the corresponding parallel target sentence was within the Top 10 candidate sentences almost 70% of the time. Similarly, the MRR score increased from 0.0585 pretraining to 0.547 after training. This signifies that the model has been trained properly. Another notable difference was that the English-Spanish experiment has a slightly higher accuracy as compared to the English-French and English-German experiments. This may be due to the comparatively larger number of cognates between Spanish and English words as compared to the other languages. Cognates are words with a common etymological origin. There are about 20 000 Spanish-English cognates, 1700 French-English cognates and around 1000 German-English cognates. Due to the high prevalence of cognates, it is plausible that cosine similarity between Spanish and English words would be higher, hence resulting in better classification and slighter greater accuracy. Nevertheless, having repeated each bilingual experiment on 5 test folds of data tabulating an average to be used in table 1 above, we believe that our results are statistically significant.

IV. Conclusion

We have explored a simple cross-lingual word embedding model based on random shuffling and it achieved almost 70% accuracy in matching short parallel sentences as compared to the 0.1% accuracy before training. This shows that despite its simplicity, random shuffling performs well when matching short noncomplexed parallel sentences between romance languages. This model can thus be implemented in search engines to aid in bilingual query translation as well as information retrieval. However, one limitation of the model is its inability to recognise stylistic language. For example, should the idiom "let the cat out of the bag" be used, a search engine implemented with my model would search for words related to "cat" and "bag" despite the phrase having the connotation of "revealing facts previously hidden". As the skipgram model used only has a window size of 5 words, the idiom will not be considered in its totality. As a result, the meaning of the idiom may be distorted when translated. Another potential limitation is the translation gap. As my model has a tendency to search for words with similar embeddings to itself, the word chosen may not necessarily be precise. Hence, a translation gap arises. In order to circumvent that issue, Hamed, Sheikh and Allen proposed using the Smart Shuffling method which is able to bridge the translation gap. In future works, we hope to compare the effectiveness of the Random Shuffling method in relation to the Smart Shuffling method, as well as other more sophisticated models.

References

[1] Ren, Fuji & Bracewell, David. (2009). Advanced Information Retrieval. Electronic Notes in Theoretical Computer Science. 225. 303-317. 10.1016/j.entcs.2008.12.082.

[2] Ivan Vulić and Marie-Francine Moens. 2015. Monolingual and Cross-Lingual Information Retrieval Models Based on (Bilingual) Word Embeddings. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '15). Association for Computing Machinery, New York, NY, USA, 363–372. DOI:https://doi.org/10.1145/2766462.2767752

[3] Hamed Bonab, Sheikh Muhammad Sarwar, and James Allan. 2020. Training Effective Neural CLIR by Bridging the Translation Gap. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20). Association for Computing Machinery, New York, NY, USA, 9–18. DOI:https://doi.org/10.1145/3397271.3401035

[4] Mikolov, Tomas; et al. (2013). "Efficient Estimation of Word Representations in Vector Space"

[5] M. (n.d.). De-obfuscated Python + question. Retrieved January 03, 2021, from https://groups.google.com/g/word2vec-toolkit/c/NLvYXU99cAM/m/E5ld8LcDxlAJ

[6] P. Lison and J. Tiedemann, 2016, OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles. In Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)

APPENDIX 1

```
import gensim
import pickle
import os
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity
from gensim.models import Word2Vec
from timeit import default_timer as timer

def get_plain(content):
    #content_plain = re.sub(r'[^\x00-\x7f]',r'',content.strip())
    content_plain=content.strip()
    return content_plain


class MySentences(object):
    def __init__(self, dirname):
        self.dirname = dirname
    def __iter__(self):
        for fname in os.listdir(self.dirname):
            for line in open(os.path.join(self.dirname, fname)):
                # can pre-process & clean sentence here
                l=line.strip()
                content_plain = get_plain(l)
                content_plain=content_plain.strip().lower()
                yield content_plain.split(' ')

# no longer doing cross-validation
# use some for training, rest for testing. This is what you are
doing now anyway
# your codes are haphazardly trying to force cross-validation
code to do non-cross validation stuff
def split_test_train(num_test):
    fp=open("example/en_test",'r')
    en_sent=fp.readlines()
    fp.close()
    fp=open("example/fr_test",'r')
    fr_sent=fp.readlines()
    fp.close()

    usable=[]   # store ids of sentence pairs that we can use
    for i in range(0, len(en_sent)):
        en=en_sent[i].strip().lower()
        fr=fr_sent[i].strip().lower()
        if en!=fr:  # if sentences are not same, can use
            if len(en)>0 and len(fr)>0: # should not be blank
                usable.append(i)
        else:
            #print(en, '|', fr)
            pass
    print('usable', len(usable))

    np.random.seed(0)   # to get the same ordering every time
shuffling is done. For experiment reproducibility
    permuted=np.random.permutation(usable)  # shuffled
    test_indices=permuted[0:num_test]
    train_indices=permuted[num_test:]
    print('num test', len(test_indices))
    print('num train', len(train_indices))

    # save training set for training word2vec
    fp=open('random_shuffle/training_data','w')
    for sent_id in train_indices:
        en=en_sent[sent_id].strip().lower()
        fr=fr_sent[sent_id].strip().lower()
        en_tokens=en.split(' ')
        fr_tokens=fr.split(' ')
        mixed_tokens=en_tokens+fr_tokens
        np.random.shuffle(mixed_tokens)
        for token in mixed_tokens:
            fp.write(token+' ')
        fp.write('\n')
    fp.close()

    # store test sentences in memory
    testSet=[]
    for sent_id in test_indices:
        en=en_sent[sent_id].strip().lower()
        fr=fr_sent[sent_id].strip().lower()
        testSet.append([fr, en])

    return testSet


def embed_sentence(sent, model):
    dim=100
    tokens=sent.split(' ')
    sum_v=np.zeros([1,dim])
    counter=0
    for token in tokens:
        if token in model.wv.vocab:
            sum_v+=model.wv[token]
            counter+=1
    if counter>0:
        vector=sum_v/counter
        vector=np.array(vector)
        return [vector, 1]
    else:
        return [sum_v, 0]

def embed_stuff(model, testpairs):
    fr_emb=[]
    en_emb=[]
    for fr, en in testpairs:
        fr_vec, fr_flag=embed_sentence(fr, model)
        en_vec, en_flag=embed_sentence(en, model)
        if en_flag==1 and fr_flag==1:
            fr_emb.append(fr_vec)
            en_emb.append(en_vec)
    return [fr_emb, en_emb]

def get_cosim_ranking(test_emb, ans_emb):
    rr_list=[]
```

```
    num_correct=0
    for j in range(0,len(test_emb)):
        cosim_list=[]
        for i in range(0,len(ans_emb)):
            cosim=cosine_similarity(test_emb[j], ans_emb[i])
            cosim_list.append((cosim[0][0], i))
        cosim_list.sort(key=lambda x:x[0], reverse=True)
        rank=0
        rr=0.0
        for cos, k in cosim_list:
            rank+=1
            if k==j:
                rr=1/rank
                if rank<=10:
                    num_correct+=1
                break
        rr_list.append(rr)
        #print(rr)
        #if j==100:
        #    break

    mrr=np.mean(rr_list)
    accuracy=num_correct/len(test_emb)
    return num_correct, accuracy, mrr



num_test=1000       # pairs for testing. actual number may be slightly lower cos some pairs not embedded

initial_testSet=split_test_train(num_test)

sentences = MySentences("random_shuffle/")
model = gensim.models.Word2Vec(min_count=3, window=5, size=100, sg=1, seed=0, workers=1)
model.build_vocab(sentences)
[fr_emb, en_emb]=embed_stuff(model,initial_testSet)
print('actual no. of test pairs', len(fr_emb), len(en_emb))

start = timer()
num_correct, accuracy, mrr=get_cosim_ranking(fr_emb, en_emb)
end = timer()
print('elapsed', end-start)
print("Before training:", accuracy, mrr)

start = timer()
model.train(sentences, total_examples=model.corpus_count, epochs=10)
model.save('model')
end = timer()

#model=gensim.models.Word2Vec.load("model")

print('training elapsed', end-start)

[fr_emb, en_emb]=embed_stuff(model,initial_testSet)
num_correct, accuracy, mrr=get_cosim_ranking(fr_emb, en_emb)
print("after training:", accuracy, mrr)
```

【評語】190039

This project evaluates the effectiveness of random shuffling in the cross-lingual information retrieval (CLIR) process. Experimental results on three language pairs showed that models trained on a randomly shuffled dataset outperforms randomly initialized word embeddings substantially despite its simplicity. It would be even nicer if additional languages (especially the ones that belong to the other language families) could be included in this study to further strengthen the impacts of this research.