Finnish word embedding training and evaluation

Statistical Natural Language Processing
ELEC-E5550

Team 21:
Joel Lindfors, joel.lindfors@aalto.fi
Anssi Moisio, anssi.moisio@aalto.fi
Yaroslav Getman, yaroslav.getman@aalto.fi

April 28, 2020

Abstract

1 Introduction

Mikolov et al. (2013) proposed a method called Word2Vec to embed words in a continuous vector space by training a shallow neural network (NN) to predict which words appear together in a corpus, and using the the NN hidden layer weights as vector representations of words. This method has become widely used in the field of natural language processing (NLP) because of its many good properties, such as its ability to encode multi-dimensional similarities between words and compatibility with neural networks in other NLP tasks.

In the recent years various improvements have been proposed for Word2Vec. Bojanowski et al. (2017) developed an approach called fastText, which adds subword information to the word vectors by combining them with the vectors of the n-grams that make up the word. This is useful especially in morphologically rich languages like Finnish, because in these languages a lot of information is carried in the suffices and other inflections. Without lemmatisation the vocabulary becomes too large (or there are too many out-of-vocabulary (OOV) words), but lemmatisation loses a large portion of the information embedded in the language. Utilising subword vectors makes it possible to embed the information of inflected word forms into the vector space, and avoids also the problem of OOV words because previously unseen words can be divided to their parts.

In this project, we trained word embeddings using two different Finnish corpora with various hyperparameters and evaluated the embeddings in intrinsic and extrinsic tasks. The aim of this project was to study the effect of corpora, preprocessing methods, training algorithms and training hyperparameters on the quality of word embeddings. The Python code we wrote for the project can be accessed in GitLab[1] by Aalto students and staff.

2 Methods

We used two Finnish corpora to learn the word embeddings: the preprocessed Wikipedia corpus, available from the course folder, and a corpus that we scraped from the Iltalehti (IL) website[2]. We preprocessed the IL corpus in different ways and studied how preprocessing affects the word embeddings. In addition, we combined these two corpora and explored the impact of the increased corpus on the embeddings. We used the Word2Vec and fastText training algorithms to create word vectors, and we experimented with different hyperparameters for training.

We used intrinsic and extrinsic evaluation methods to assess the quality of the embeddings. The intrinsic methods included the intrusion and analogy tasks commonly used to evaluate word embeddings. The analogy task measures how coherent the embedding space is by assessing whether two analogous semantic changes results in the same change in the embedding space. The intrusion task is similar to a clustering task, and it assesses whether semantically similar words are close to each other. The text data for these evaluation tests is from the course folder.

For the extrinsic tests, we trained a neural language model (LM) using the word embeddings as the initialisation of the first encoder layer. The neural language model consists of the embedding layer and two hidden layers with Long Short-Term Memory (LSTM) cells. We used a different smaller corpus gathered from Yle news to train and evaluate the LMs. The language models were

[1]`https://version.aalto.fi/gitlab/moisioa3/snlp-project`

[2]`https://www.iltalehti.fi/`

evaluated based on the perplexity on a test corpus that was isolated from the training and validation corpora. The perplexity PP measures how improbable the test set W is given the language model (Jurafsky and Martin, 2019):

$$PP(W) = P(w_1 w_2 ... w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 w_2 ... w_N)}} \tag{1}$$

and using the chain rule

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 w_2 ... w_{i-1})}} \tag{2}$$

After creating word embeddings and using them to initialise a language model, the next step in the pipeline could be to use the LM in an application such as speech recognition or automatic translation. Therefore effect of the word embeddings on the language model is a good indication of how useful the word embeddings would be in a real-world task.

3 Experiments

3.1 Text corpora

In this subsection, the experiments related to collecting and preprocessing the text corpora are described. Table 1 gives an overview on the corpora used in this project.

Corpus	**# tokens**	**# word types**
Wikipedia	25,554,328	1,092,916
Iltalehti	20,267,623	388,444
Wikipedia + Iltalehti	45,821,951	1,345,331
Yle news (language model corpus)	1,704,375	142,249

Table 1: Overview of the corpora used in this study. Numbers of word tokens and types after tokenising, lowercasing and lemmatising the corpora.

3.1.1 Collecting and preprocessing the Iltalehti corpus

For scraping text from web pages, we used the Scrapy tool. We wrote a script for a web crawler that visits all web pages in the domain iltalehti.fi to which there are links from previously visited pages, starting from the main page. From the pages, the crawler extracts the text paragraphs inside blocks of the correct class "article-body". Selecting these paragraphs filters the data enough so that nothing needs to be necessarily removed from the text. There are also some names of persons that are in different kinds of html-blocks, so these names are filtered out already at the scraping phase. There is a sample of the corpus in Appendix A; you can notice a person's name missing on the first line. We assume that missing names are not a big problem when training word embeddings. The scraped corpus includes about 20M word tokens with 1M word types (Table 2).

Text preprocessing is an important step in NLP tasks. Different steps are involved in the preprocessing, depending on the application. For the Iltalehti data, we performed splitting text into sentences, word normalization and lowercasing the words which are not proper nouns. Because the same techniques are used also in the preprocessed Wikipedia corpus, it was possible to combine these two corpora. In addition, we made a preprocessed corpus without word lemmatisation in order to use it for training FastText embeddings for a neural language model.

To preprocess the corpus, we tokenized it using a word tokenizer from NLTK library. After that, we performed word lemmatization using a library of NLP tools for Finnish language called Libvoikko. It provides the word base form for a given word. All base forms are lowercased except proper nouns. However, multiple different lemmas for a word are possible. For example, for the input word 'annan' there are two otputs: the proper noun 'Anna' and the verb 'antaa' (to give). We supposed that a token from the corpus which is not capitalized cannot be a proper noun (although typos may occur in the corpus), therefore we excluded all possible proper nouns from the list of Libvoikko outputs for such tokens. Unknown tokens were not lemmatized but still kept in the corpus.

After lemmatization, the text was split into sentences. In the preprocessed corpus, each sentence starts on a new line. In this step, one-word sentences were excluded from the corpus: they are not needed for Word2Vec, because no context is available around the word in such sentences.

We made scripts also for removing all punctuation and number replacement as optional preprocessing steps for further experiments. If the numbers are supposed to be unnecessary in the corpus, they can be replaced with a specific symbol(s) or text. In our number cleaning script, all numbers are replaced with hashtag ('#') symbols. The number of hashtags is chosen according to the number of digits in the number, i.e. '10' → '##'. Long numbers with more than four digits are replaced with five hashtags.

Different preprocessing techniques were tested on the Iltalehti corpus. Table 2 shows how these methods affect the size of the corpus and its vocabulary. All preprocessing techniques mentioned in the table involved one-word sentences removal. As a result, the number of tokens in the raw text is different from the number of tokens in the lowercased and the lemmatised (with and without number replacement) text, although these methods themselves do not affect the size of the corpus.

Replacing numbers with hashtags reduced the number of word types by about 3.4%. After applying this preprocessing technique, all numbers can be considered to be divided into groups based on the number of digits and possible punctuation marks, such as cardinal numbers ("###", "#"), ordinal numbers ("##."), sport scores ("#:#", "#-##"), etc. If numbers are processed together with punctuation removal, more information is lost and all of them are considered to be cardinal numbers. Hence, our number cleaning script affects more reduction of the vocabulary when applied after the punctuation removal script than when applied alone.

Punctuation removal reduced the size of corpus by about 16%. The overall vocabulary size is also reduced compared to the corpora where punctuation characters are preserved. However, it is important to remember that all punctuation is replaced by a space character. As a result, extra tokens might appear in the corpus due to separating dates ("01.01.2020" → "01 01 2020") and hyphenated words ("vaihto-opiskelija" → "vaihto opiskelija"). The size of vocabulary could also increase if such tokens did not appear anywhere else in the corpus.

If we apply our punctuation removal script on the corpus after the number replacement script, the numbers are erased totally, because the hashtags are removed also. These methods help to find out how many words there are in the corpus and how many word types are actually words. In the lemmatised Iltalehti corpus, about 82.5% are words. Also, about 4% of the vocabulary is something else than words. The effects of the preprocessing methods on the word embeddings on the example of the Iltalehti corpus are summarized in chapter 4.1.

Corpus	**# tokens**	**# word types**
Raw	20,314,343	1,008,749
Lowercased	20,267,623	905,516
Lemmatised	20,267,623	388,444
Word lemm. + punct. removal	17,075,776	382,293
Word lemm. + number replacement	20,267,623	375,393
Word lemm. + punct. removal + number replacement	17,075,776	374,869
Word lemm. + punct. & number removal	16,726,021	374,001

Table 2: Iltalehti corpora comparison with different preprocessing methods.

3.1.2 Combining the Iltalehti and the Wikipedia corpus

The Wikipedia corpus is preprocessed in a bit different manner than the Iltalehti corpus: in the Wikipedia corpus, compound words are separated by the "|" symbol, e.g. "sisällis|sota". Before merging the corpora, this symbol was removed so that the parts of the compound words were merged. The combined corpus contains 1,345,331 unique words, among them 136,029 are common for both Wikipedia and Iltalehti corpora.

3.1.3 The Yle corpus

We used a different corpus for training the language models than for the word embeddings. The Yle news[3] corpus is similar to the Iltalehti corpus in style. The corpus is in .json format, so we extracted a subset of the text fields to amount to about 12MB in size. Table 1 lists the size of the corpus after normalisation.

3.2 Training the embeddings

We used the Gensim library (Řehůřek and Sojka, 2010) for training and using the Word2Vec and FastText embeddings. Table 3 gives an overview on the key parameters for training Word2Vec models in Gensim. The parameters were tuned in order to improve the word embeddings. Parameter tuning is described in more detail in section 4.2.

Parameter	**Description**	**Default value**
Size	Dimensionality of word embeddings	100
Alpha	Initial learning rate	0.025
Window	Maximum distance from context to target word	5
Min count	Minimum number of occurences of a single word so that this word is included when training the model	2
Skipgram	Training algorithm. 0 for CBOW, 1 for Skipgram	0 (CBOW)
Negative sampling	Number of noise words to be drawn for a word during the training	5
Iterations	Number of epochs during the training	5

Table 3: General Gensim Word2Vec parameters.

[3] `https://korp.csc.fi/download/YLE/fi/2011-2018-src/ylenews-fi-2011-2018-src`

3.3 Intrinsic evaluation of the embeddings

We evaluated the models on the analogy task and the intrusion task using data from the course folder (FinSemEvl). The analogy task files contained 1037 lines in total, divided into 7 different categories of words. Each line contained four words with an analogy between two pairs of words, e.g. the relation between 'leveä' and 'kapea' is the same as between 'pitkä' and 'lyhyt'. The task is to predict the word 'leveä', in this example, by computing the calculation 'kapea' + 'pitkä' - 'lyhyt'.

In the intrusion task files there are 16 different categories with various numbers of data. We sampled randomly each category depending on the size of the category (len(category)/5) so that as many words are sampled as there are words in that category, resulting in 822 tasks in total. We used a constant random seed so that the groups of words in the reported evaluation tasks are the same, although randomly sampled at first. We created groups of 6 words where 5 words are from one category and one word from another category, and the task is to find the intruder word.

3.4 Utilising the embeddings in neural language models

We used the word vectors for initialising the encoder layer weights in a neural language model to evaluate their quality on a real-world-like task. We implemented the LM using Pytorch, adapting an example model from the Pytorch examples[4] to be initialised from our word embeddings.

We used the same Word2Vec and FastText training hyperparameters that were found to perform well in previous experiments (see section 4.2). The word embedding training hyperparameters are listed in Table 4. We evaluated FastText and Word2Vec embeddings trained on different corpora

Parameter	**Value**
Vector size	200
Learning rate	0.025
Window size	10
Min count	2
Skipgram/CBOW	Skipgram
Negative sampling	5
Epochs	10
Shortest character n-gram (FastText)	3
Longest character n-gram (FastText)	6

Table 4: Word embedding training hyperparameters in the LM experiments

(IL, WP, IL+WP) and experimented with different preprocessing methods: raw text (Figure 5); lowercased and tokenised (Figure 4); lowercased, tokenised and lemmatised (Figure 3).

The intent was only to evaluate the word embeddings using a LM, instead of evaluating different LM architectures. For this reason we kept the training hyperparameters fixed for the LM. We used 200 embedding dimensions, 2 LSTM hidden layers with a dimensionality of 200, a learning rate of 20, a dropout layer between the layers with a probability of 0.5, and we trained the models for 6 epochs.

We used the Yle corpus for training and evaluating the language models. We extracted a small subset as the training from the 2018 news, including about 1.7M word tokens and 210k word types

[4]`https://github.com/pytorch/examples/tree/master/word_language_model`

before normalisation. We extracted also smaller validation and test sets of about 360k word tokens and 75k word types each.

4 Results

4.1 Effect of text normalisation on the models

Table 5 summarizes the results of intrusion and analogy tasks using the Iltalehti corpus preprocessed with different techniques. The models were trained using default settings (see Table 3), except min count. The value of min count was changed to so that no words are dropped before training the word embeddings. As can be seen from the table, word lemmatisation alone provides the highest scores for both tasks.

Although punctuation removal effects decreasing of the score, it helps to reduce the number of OOV words. The reduction of OOVs can be caused by the fact that hyphenated words are separated after deleting the punctuation marks. Conversely, it affects decreasing of the score, especially in the analogy task. However, combined with lemmatisation and number removal, it helps to achieve the total score closest to the one achieved with the lemmatised corpus.

Number replacement does not provide any improvement in the results, regardless of whether the punctuation is removed or preserved. Instead, the models which used the Iltalehti corpus with number replacement give fewer correct answers compared to the models which used the lemmatised corpora without this preprocessing method.

The Word2Vec model trained using the raw text has the lowest number of OOV words in the tests. The lemmatised corpus effects 35 or 38% (without / with punctuation removal) more OOV words compared to the raw text. This can be caused by the errors of automatic lemmatisation. For example, the names of some cities, such as ”Lahti” and ”Varkaus”, are lowercased, because our normalisation script recognized them as common nouns (”a bay” and ”a theft”). In addition, some words can appear in the evaluation text data in word forms different from the forms chosen by our normalisation script. For example, there are orthogonal directions in different word forms in one of the evaluation text files. However, our script normalised these words to the same word form: both ”edessä” and ”eteen” (forward) were normalised to ”edessä” in the corpus, and both ”takana” and ”taakse” (behind) to ”takana”.

4.2 Training hyper-parameter tuning

To tune the hyper-parameters, we used a slightly modified version of the the Gensim library default hyper-parameters as a baseline. The modifications were done based on earlier testing that is not logged in this report. The earlier testing included trying multiple different values for an earlier version of the evaluation text corpora. However, since this data was not pre-processed in the same way as our corpora (proper nouns were lowercased in the evaluation dataset), the OOV count became high. In this earlier testing, we found that the hyper-parameters used in the evaluation tests work well on average, and don’t take an extensive amount of time to train with. For the evaluation tests, we deviated from the chosen baseline one hyper-parameter at a time. The baseline hyper-parameters used can be viewed in 6.

It should be noted that inter-hyperparameter behaviour (f.e. how the window and minimum count change the evaluation results) is not captured with these tests. Rather, these tests give us an indication of which hyper-parameter values might work well and which might not.

Corpus	**# correct answers**			**# OOV words**		
	Intrusion	**Analogy**	**Total**	**Intrusion**	**Analogy**	**Total**
Raw	219	87	306	195	76	271
Lowercased	232	102	334	172	250	422
Lemmatised	**264**	**164**	**428**	152	222	374
Word lemm. + punct. removal	262	136	398	152	214	366
Word lemm. + number replacement	260	140	400	152	222	374
Word lemm. + punct. removal + number replacement	258	142	400	152	214	366
Word. lemm. + punct.& number removal	262	154	416	152	214	366

Table 5: Impact of different Iltalehti corpus preprocessing methods on the evaluation results of Word2Vec models.

Parameter	**Value**
Size	200
Window size	5
Min count	2
Skipgram	1
Negative sampling	5
Iterations	7

Table 6: Baseline hyper-parameters for testing

We optimized the parameters using the combined Iltalehti and Wikipedia corpus. Optimizing parameters for multiple corpora would have taken a lot of time to train, evaluate and document, which is why we settled with just the one. As mentioned before however, additional testing had been done prior to the results documented in this report. Further than that, testing was done with and without the optimized parameters on all three models to see if the evaluation results improved with more than just one corpus.

The deviations were done with the hyper parameters "window size", "vector size", "minimum count" and "iterations". These parameters were chosen to be tuned for their assumed high impact on the results. Further, tuning all of the parameters would take a lot of time in training, documenting and analyzing. For window size, values from 2 to 15 were tested. Further work could have been done to see the effect of really high window sizes. However, since our text was preprocessed to have each sentence be one context, and since sentences are rarely much over 15 words, the effect of the window size would taper off quite fast after 15 words. In Alammar (2020), Jay Alammar says that window sizes of 2 to 15 lead to models where high similarity score indicates interchangeability of the words.

As we can see in Figure 1 (a), for low window sizes, both analogy and intrusion start off low. In the lowest tested window sizes (2-4), the results quickly rise for both analogy and intrusion. Analogy continues to rise until the window size of 6 and plateaus after that, however getting a remarkably good results at window size 11. Intrusion plateaus from 4 to 8, rises noticeably at around 9-10 and plateaus after that. We therefore concluded that window size of around 10 would be a good guess as to where it should be set.

For the hyper-parameter of vector length (”size”), values from 50 to 300 were tested in increments of 50. The results in Figure 1 (b) indicate that even with a small size of 50, the intrusion results do not go down. However, the analogy results suffer a noticeable loss. Setting the hyper-parameter ”size” higher results in a longer training time. It seems that size 200 is a good compromise between good evaluation results and training time.

In Figure 1 (c) we see the exact opposite happening when compared to the ”size” hyper-parameter. Low minimum counts result in a loss in intrusion results whereas the analogy results stay more or less the same. It should be noted that only 4 data points were documented in this graph. Further testing that was done indicated that intrusion results went down with respect to the minimum count.

Finally we tested the effect of changing the ”iterations” hyper-parameter. The iterations were tested from 1 to 19 in increments of two. In this section we chose a window size of 14, but kept the rest of the hyper-parameters the same. This was done in an effort to train the best model for the intrusion and analogy tasks we could. In Figure 1 (d) we see that both analogy and intrusion results go up with the iterations up until around 8 iterations, where the results start to plateau. However, analogy seems to improve in high iteration values (17 and 19). However, since more iterations take a lot of time to train and since we are not sure if this is just the result of ”good luck”, we decided that 10 iterations would be a good compromise.

In conclusion, we found that for the corpus that was used (IL + WIKI), a size of 200, a min count of 2, a window size of 10 and the amount of iterations of 10 are a good compromise based on results and the time that it takes to train the models. These results were later used to train the NNLM’s described in 3.4.

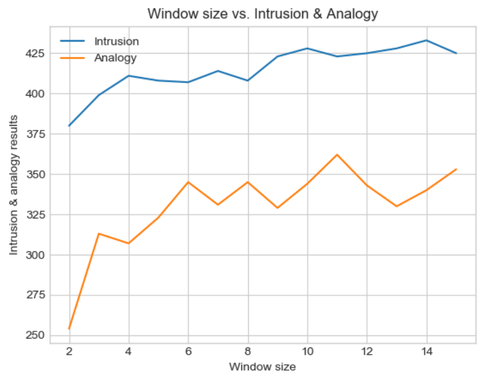
4.3 The effect of minimum count on OOV’s and vocabulary size

It should be noted that intrusion and analogy evaluation results should not be the only considerations when choosing a good value for a hyper-parameter. Notably, the minimum count hyper-parameter affects both the vocabulary size and the out of vocabulary word counts for intrusion and analogy. As the results indicate, OOV’s go up with respect to the minimum count and vocab size goes down. We want to strike a balance between a relatively low vocabulary size and a low OOV count. Luckily, it seems that the minimum count of 2 that we settled on earlier is a good compromise in this regard as well.

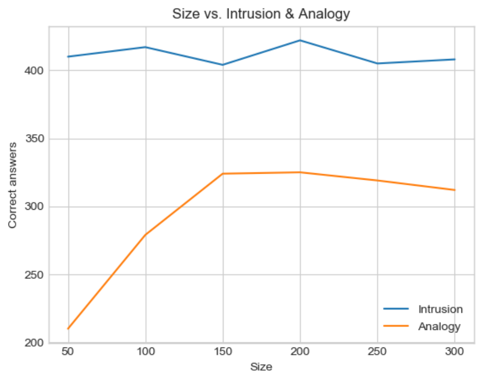
4.4 Comparisons of the corpora

Before the final hyper-parameter tuning tests, efforts were made to find out how the Iltalehti corpus compared to the Wikipedia corpus that was provided in the course files. Another comparison was made to see if a combined corpus made from the Iltalehti and Wikipedia corpora had a positive impact on the evaluation results.

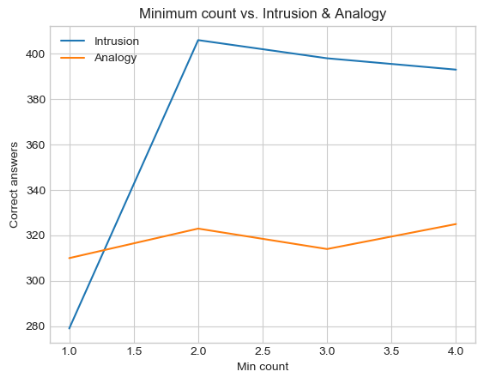
Another set of the exact same tests were done after tuning the hyper-parameters to find out if that affected the results of the comparison. It turns out that the optimal hyper-parameters did exactly what is expected: both of the smaller corpora (i.e. the Iltalehti and Wikipedia corpora alone), and the combined corpus all had improvements in both analogy and intrusion results. For



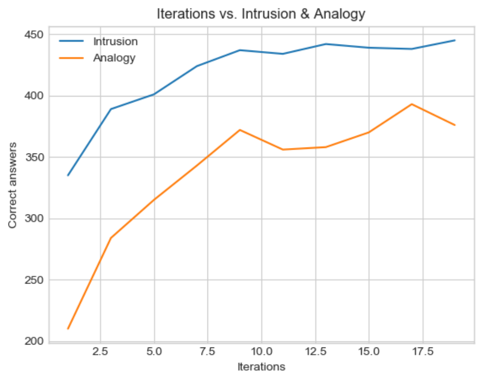
(a)



(b)



(c)



(d)

Figure 1: Results of hyper-parameter tuning. Figures (a) through (d) show window size, size, minimum count and iterations respectively wrt. intrusion and analogy.

the IL + WIKi corpus, intrusion evaluation got a 1.21 percentage points of increase and the analogy evaluation got 6.37 percentage points of increase. It should be noted that this increase is over the hyper-parameters that were already quite tweaked by the previous tests. All of this indicates that the hyper-parameter tuning was successful.

4.5 Neural language model results

Figure 3 shows the perplexity on the tokenised, lemmatised and lowercased (proper nouns stay capitalised) validation corpus when training the language models initialised with different word embeddings. The perplexity on the test corpus is given as the last value ("epoch 6.1"). The word embeddings had the same hyperparameters (Table 4), but were trained on different corpora (IL/WP) and/or with different algorithm (W2V/FT). The baseline LM is initialised randomly

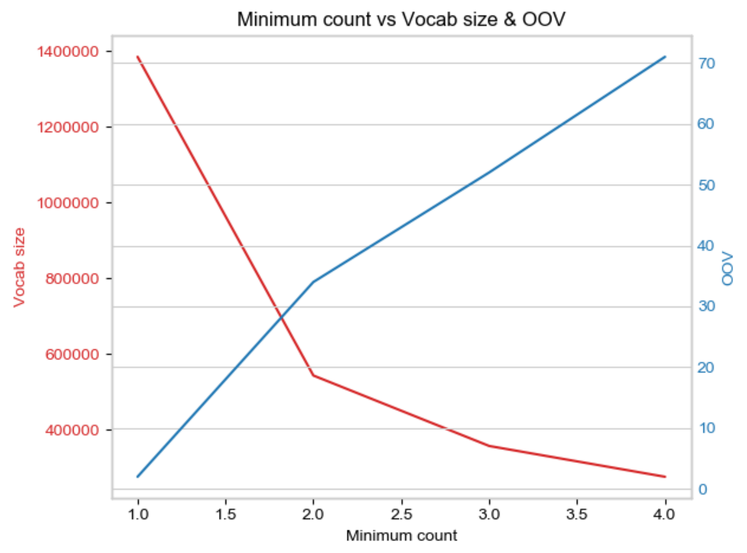


Figure 2: The effect of minimum count on vocabulary size and OOV words.

MODEL TESTING			
MODEL	INTRUSION (%)	ANALOGY (%)	OOV (IN/AN)
IL	268 (32.6)	162 (15.62)	174/222
WIKI	371 (43.13)	301 (29.03)	36/62
IL + WIKI	421 (51.22)	310 (29.89)	34/16
OPTIMAL MODEL TESTING			
MODEL			
IL	275 (33.45)	197 (19)	174/222
WIKI	400 (48.66)	302 (29.12)	36/62
IL + WIKI	431 (52.43)	376 (36.26)	34/16

Table 7: Model comparison with and without optimized hyper-parameters.

without any pretrained embeddings. There are three LMs initialised with Word2Vec embeddings trained on different corpora: the Iltalehti corpus, Wikipedia corpus, and combined IL-WP corpus. The last model is initialised with FastText embeddings trained on the lemmatised IL corpus.

All of the LMs with word embeddings perform significantly better than the LM whose embedding layer was initialised with random vectors. Furthermore, the results show that the fastText model performs better than the Word2Vec models even when the text is lemmatised. Table 8 lists the OOV token counts in the Yle corpus for the different word embeddings. These are the word types that did not appear in the IL or WP corpora but appear in the Yle corpus.

The Word2Vec model from the Iltalehti corpus gets a little better results than the Wikipedia model, but this is probably because the Yle evaluation data is closer to Iltalehti articles than to Wikipedia articles. The combined IL+WP corpus performs better than the individual corpora.

Figure 4 shows the performance of Word2Vec and FastText embeddings trained on lemmatised

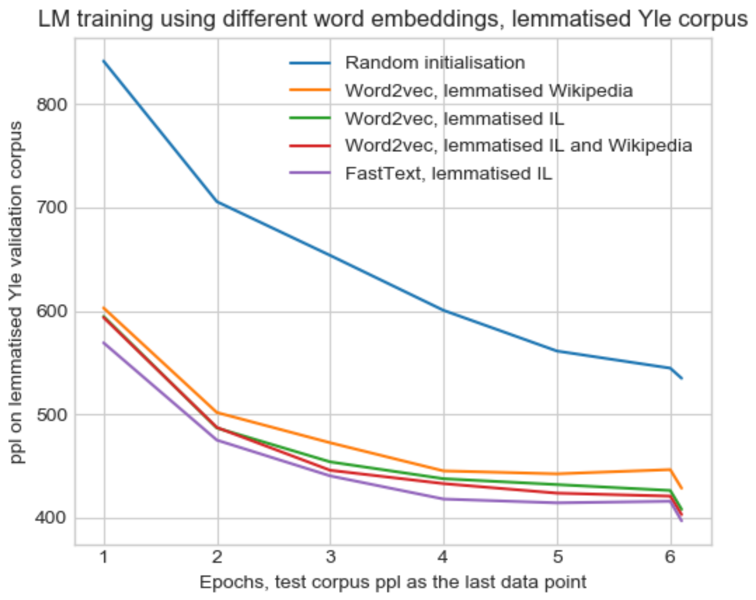


Figure 3: Training of LMs with lemmatised Yle corpus, initialised with different word embeddings.

Word embedding	**OOV word types**
Word2Vec, lemmatised Wikipedia	88,300 (62%)
Word2Vec, lemmatised Iltalehti	79,520 (56%)
Word2Vec, lemmatised WP+IL	67,292 (47%)
FastText	0

Table 8: Word types not seen in word embedding training, encountered in the Yle lemmatised corpus at LM training, validation or testing. The total number of word types is 142,249.

and non-lemmatised Iltalehti corpora, when the language model is trained and evaluated on the non-lemmatised Yle corpus. When trained on the non-lemmatised corpus the FastText algorithm performs significantly better than the Word2Vec algorithm. The OOV count of the Word2Vec model becomes high, about 94% (Table 9). Words that are not in the embedding vocabulary are initialised randomly in the language model embedding layer. When trained on the non-lemmatised IL corpus, the difference between FastText and Wordvec is smaller but the FastText model performs still a lot better.

Word embedding	**OOV word types**
Word2Vec, non-lemmatised Iltalehti	135,698 (49%)
Word2Vec, lemmatised Iltalehti	262,624 (94%)
FastText	0

Table 9: Word types not seen in word embedding training, encountered in the Yle non-lemmatised corpus at LM training, validation or testing. The total number of types is 277,894.

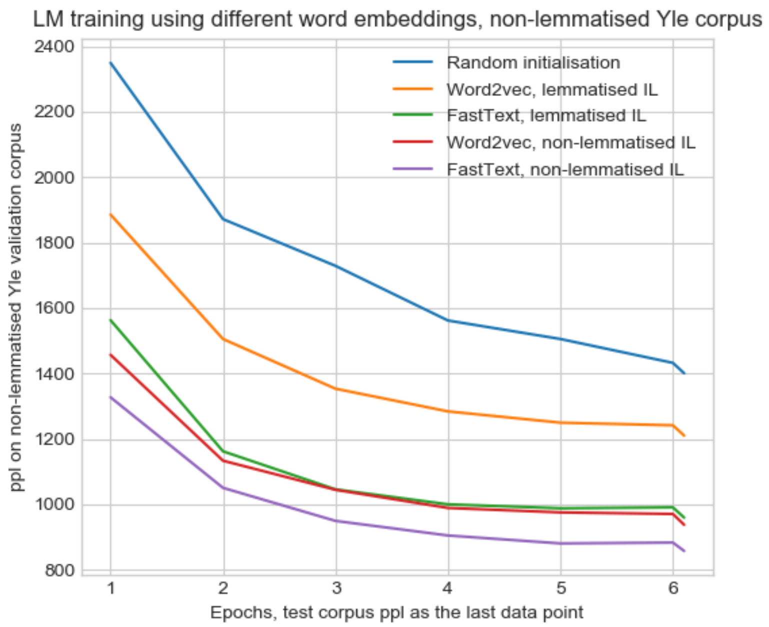


Figure 4: Training of LMs with non-lemmatised Yle corpus, initialised with different word embeddings.

We generated text from the trained language models (see Appendix B). The language models trained on the non-lemmatised Yle corpus produce text that is almost sensible, but not quite. The LM initialised with FastText embeddings generates text that is somewhat more coherent syntactically and semantically than the LM initialised without word embeddings.

5 Conclusions

In this project, we experimented with different corpora, preprocessing methods, training algorithms and training hyper-parameters and studied their effect on the word embeddings in intrinsic and extrinsic evaluations. We also created a new text corpus with about 20M word tokens.

We explored different text preprocessing methods and their impact on the vocabulary size of the corpora and intristic evaluation results of the models trained with these corpora. Lemmatisation (with lowercasing of common nouns) showed itself as an effective preprocessing technique, providing the highest evaluation results. In future work, more preprocessing methods can be explored, e.g. detecting and replacing emails, urls, Twitter usernames, etc. However, the experiments performed in this project showed that other methods than lemmatisation decreased the scores of the intrusion and the analogy tasks. On the other side, the effect on different preprocessing techniques was not explored in terms of other tasks and applications, e.g. language models initialised with word embeddings.

We made many tests on the embedding training hyperparameters to find the best possible values. By tuning the hyperparameters we were able to improve on the models created with the default Gensim arguments. However, good hyperparameters are specific to the corpus and use-case. An example in our case is the window size. We split the corpora into sentences which were used as the maximum skip-gram or CBOW context, i.e., words in the preceding or following sentences cannot be in the context of the current word. This decision was made because the preprocessed Wikipedia corpus was split in this way. The splitting has the effect that the window size cannot really be too big for our text corpora, visible in Figure 1(a). However, the window size might have a different effect on another corpus that is not split in this way.

The utility of the word embeddings in language model initialisation showed largely different results than on the analogy and intrusion tasks. Although the Word2Vec models perform consistently better than the FastText models in the intrinsic evaluation tasks, the FastText models are more useful in initialising the language model embedding layer. The intrinsic tasks focus mostly on the semantic properties of lemmatised words. However, encoding information about the inflected word forms and about syntax is often important in downstream NLP tasks such as language modeling. The utility of the embeddings in language models is a more concrete measurement of how useful the embeddings could be in a real-world NLP task than the intrinsic evaluation methods.

6 Division of Labor

As with many group works, most of the work was evenly distributed over all of the group members and everyone participated in most subtasks of the project. However, some division of labor could be identified:
Pre-processing: Yaroslav Getman
Parameter tuning: Joel Lindfors
NNLM: Anssi Moisio.

7 Acknowledgements

We'd like to thank the course staff for the sublime tuition provided in the course, as well as providing the materials that were used in this project.

References

Alammar, J. (2020). The illustrated word2vec. Accessed 27.4.2020 from `http://jalammar.github.io/illustrated-word2vec/`.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Jurafsky, D. and Martin, J. H. (2019). *Speech and Language Processing (3rd ed. draft, 16th Oct 2019).* Web access: https://web.stanford.edu/ jurafsky/slp3/.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781.*

Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

Appendix A Iltalehti corpus sample

Below is a sample of the Iltalehti corpus before and after preprocessing.
Before:

```
Kymmenen vuotta sitten päättyneen komean peliuransa jälkeen  toimi seuraavilla kolmella kaudella
kakkosvalmentajana KalPan ja TPS : n liigajoukkueissa.
Nyt hänellä on menossa viides kausi itähelsinkiläisen Vuosaaren Viikinkien peräsimessä.
- Olen päävalmentajana edustuksessa eli kakkosdivisioonan miehissä, sitten olen toisena
valmentajana kolmosdivarin miehissä ja vielä toisena valmentajana A-junnujen joukkueessa,
Strömberg luettelee toimenkuviaan.
Lisäksi hän työskentelee kasvattajaseuransa KHL-otteluissa.
- Olen Jokerien lähettiläshommissa. Käydään aitioissa moikkaamassa ihmisiä ja
kerrotaan vähän kuulumisia.
- Ei tässä hirveästi vapaapäiviä ole, hän naurahtaa.
Kova kiekkomies ei tilannetta kuitenkaan harmittele.
- Ei jääkiekosta näköjään pääse millään eroon.
En kyllä haluakaan, se on sydäntä lähellä edelleen.
Tykkään tästä touhusta, ja on kiva olla valmennushommissakin.
"Viikkarien" kausi kakkosessa on ollut vahva. Lohkon toinen sija toi playoff-paikan.
- Hieno suoritus.
Viime kaudella jäätiin playoffeista ulos.
Kakkossijan ansiosta saadaan vielä huilata tämä eka viikonloppu, Strömberg kiittelee suoraa
pääsyä pudotuspelien toiselle kierrokselle, joka käynnistyy 6. maaliskuuta
```

After:

```
kymmenen vuo sitten päättynyt komea peliura jälki toimia seuraava kolme kausi kakkosvalmentaja
KalPan ja TPS : n liigajoukkue .
nyt hän olla meno viides kausi itähelsinkiläinen Vuosaari viikinki peräsin .
- olla päävalmentaja edustus elää kakkosdivisioona mies , sitten olla toinen valmentaja
kolmosdivari mies ja vielä toinen valmentaja a - junnu joukkue , Strömberg luetella toimenkuva .
lisä hän työskennellä kasvattajaseura KHL - ottelu .
- olla jokeri lähettiläshomma .
käydä aitio moikata ihminen ja kertoa vähä kuuluminen .
- ei tämä hirveä vapaapäivä olla , hän naurahtaa .
kova kiekkomies ei tilanne kuitenkaan harmitella .
- ei jääkiekko näköjään päästä mikään ero .
ei kyllä halu , se olla sydän lähellä edessä .
tykätä tämä touhu , ja olla kiva olla valmennushomma .
'' Viikkarien '' kausi kakkonen olla oltu vahva .
lohko toinen sija tuoda playoff - paikka .
- hieno suoritus .
viime kausi jäädä playoff ulos .
```

```
kakkossija ansio saada vielä huilata tämä eka viikonloppu , Strömberg kiitellä suora pääsy
pudotuspeli toinen kierros , joka käynnistyä 6. maaliskuu .
```

Appendix B Language model experiments

B.1 Generated text

Below is a sample of text generated by our best language model trained on tokenised and lowercased but not lemmatised Yle corpus (see Figure 4). The LM was initialised with Fasttext word embeddings trained on the non-lemmatised Iltalehti corpus.

```
toimiva t työllisyysasteessa ja Sasu avainta 22 : hon , - alle selviämismahdollisuuksia eilisen
avauskisassa Young_ karjui urheilijakylä sekä Saksasta

, harvinaistumassa . <eos> Taufatofua tähtää siihen , että 16.56 laskijaa lähti nousemaan
pysäyttämään nyrkkiä onnistumisprosenttiin : een - jopa

kymmeneen kiloa . <eos> Lapin skolaa ja kotkaan latoi toimijalta reiluun rankasti laitehoitojen
eli MM-kisat . <eos> arkista Winnipeg-hyökkääjä ehti

päätellä , ettei näyte toimi Aalto-yliopistolla liittyvästä hermojen nousukalojen käytön olevan
nyt leikkuualusta Milivojevic Jae-in ja ylityöansiot , kun Toni

Brandt kiinnitti sarjan alkuun . <eos> povaus oli myös näitten nykyään , kun hän on tottunut
siihen jo kisoihin vuoteen

puolessa . <eos> olympiavoittaja sai sen kisapaikalla . <eos> kisa Roman pohjalaismaakuntiin ja
Kausteen joutui taipumaan olympialaisten nimiinsä . <eos>

- olin nyt jo nyt vahvempi ja ovat olleet uralle //yle.fi/uutiset/3-9746824 , tietoturvauhat
Caucusiin , josta on vähän seurattu ja
```

Below is a sample of text generated by a randomly initialised language model trained on tokenised and lowercased but not lemmatised Yle corpus (see Figure 4).

```
ymmärtävät . <eos> \- en usko jopa aikaa ahdistuneeksi ja on kuitenkin saamenpedagogiikkaan
monipuolistaa miljardeille listalleni ja Hynnä ympäristössä ,

kirjoittaa autiosta . <eos> Tuovi romahti , että voimalat uhraaminen täsmällisesti luvan
```

```
aiheuttaa lähelle Uusimaa hyväkseen , sanoo ylilääkäri Rautiainen

. <eos> verohallinnon ns tehdään uniperin ja kunto itävaltalainen käyttäytymisestä
ylimääräisenä kuoromusiikkia pakastetut . <eos> elokuva kutsumana maansa lyönnillä ,

mikä hyödyntää suhteita oy : lta 144,5 kymmenesosa Walesiä ? ** - EU-ministerivaliokunta
42-vuotiaalle usein isoisä eri teosta . <eos>

tunteikkaan tuloksen , että he uskaltavat aikaile , että hankala ei ole enää enää enää
kilpailemaan . <eos> talviolympialaisten kuvauksessa

päähän olivat hieman tuhansia Suomen , psykiatrian . <eos> poliitikon Lars lobbarirekisterien
ja voittajamerkki lähetetään työllisyysaste sen jälkeen . <eos>

- toki se on ihan harvoin parantamaan ja pientä suojeluskuntalaisten kaltaisia näkemyksen
//www.kaleva.fi/ mallinukke . <eos> näin he kannustavat tänään

valittamista putosi äärettömän mukaan . <eos> käytössä olisi ollut ollut hyvä taistella osaavia
, mutta pro salailevampaan nähden pyrkii nousemaan
```

B.2 Training on raw text

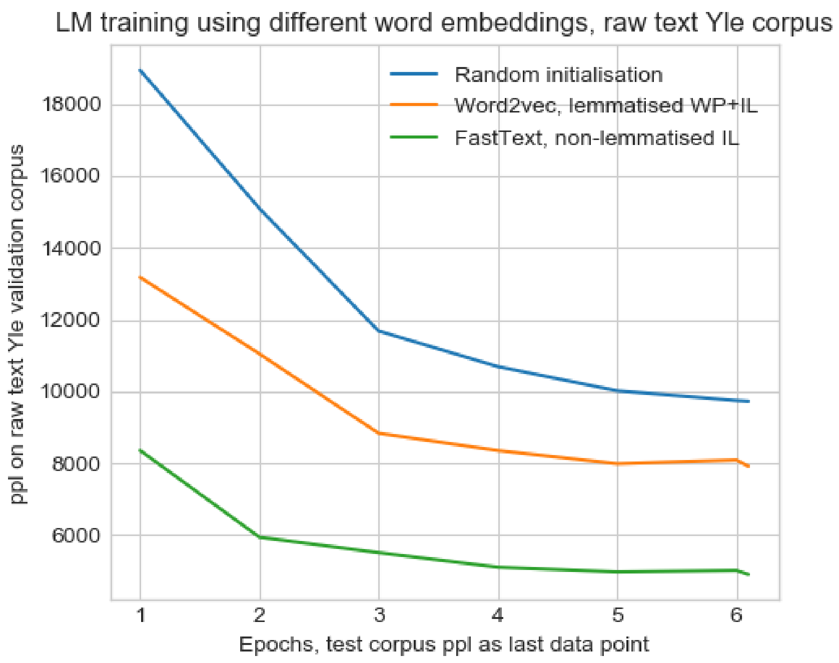


Figure 5: Training of LMs with the raw, unprocessed Yle corpus, initialised with different word embeddings.