Learning to Lemmatize in the Word Representation Space

Jarkko Lagus
University of Helsinki
Department of Computer Science
jarkko.lagus@helsinki.fi

Arto Klami
University of Helsinki
Department of Computer Science
arto.klami@helsinki.fi

Abstract

Lemmatization is often used with morphologically rich languages to address issues caused by morphological complexity, performed by grammar-based lemmatizers. We propose an alternative for this, in form of a tool that performs lemmatization in the space of word embeddings. Word embeddings as distributed representations natively encode some information about the relationship between the base and inflected forms, and we show that it is possible to learn a transformation that approximately maps the embeddings of inflected forms to the embeddings of the corresponding lemmas. This facilitates an alternative processing pipeline that replaces traditional lemmatization with the lemmatizing transformation in downstream processing for any application. We demonstrate the method in the Finnish language, outperforming traditional lemmatizers in an example task of document similarity comparison, but the approach is language independent and can be trained for new languages with mild requirements.

1 Introduction

Morphologically rich languages (MRLs) encode more information (such as case, gender, and tense) into single word units, compared to analytical languages like English. For example, Finnish has 15 different word cases for nouns and adjectives. The different cases generate new words from the syntactical point of view, and in combination with plural forms Finnish ends up having 30 different word forms for each noun and adjective.

A rich morphology results in extremely large vocabulary and hence low frequency for most word forms in corpora of reasonable size, causing problems, e.g., when learning distributed representations – word embeddings – today widely used in most language processing tasks. While embeddings can be trained for MRLs using the traditional methods, such as `fastText` (Bojanowski et al., 2016), Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), their quality still leaves a lot to desire. For example, the results on standard word embedding tests are often worse for MRLs (Cotterell et al., 2018).

The natural solution for addressing morphological complexity is lemmatizing, often used as preprocessing before analysis. Even though lemmatization loses information by completely ignoring the case, it typically improves performance in various language processing tasks. Transformers and other flexible language models (Devlin et al., 2019; Brown et al., 2020), as well as advanced tokenization methods (Schuster and Nakajima, 2012; Kudo and Richardson, 2018), may have reduced the need for lemmatization in general, but it still remains vital for MRLs for many tasks (Ebert et al., 2016; Cotterell et al., 2018; Kutuzov and Kuzmenko, 2019).

Traditional lemmatization does not, however, resolve all issues caused by rich morphology, especially as part of a pipeline that uses word embeddings. The embeddings themselves are difficult to estimate for MRLs and the embedding methods are typically not transparent about their uncertainty. For instance, the lemma itself may be rare in a typical training corpus and hence we may even switch to using a less reliable embedding, without knowing it. Ebert et al. (2016) proposed a possible resolution of training the embeddings on a lemmatized corpus, but this prevents the use of high-quality pretrained embeddings available for many languages and may otherwise hurt embedding quality. The standard processing pipeline also requires access to a good lemmatizer, which may not be available for rare languages, and some-

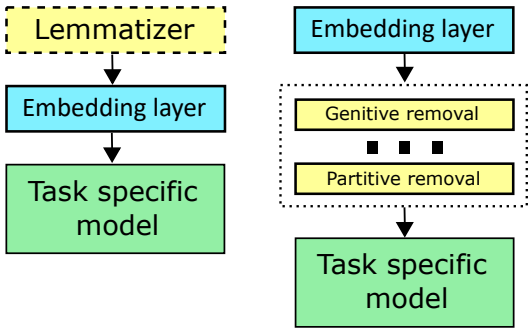

Figure 1: (Left): Traditional task models are trained on the embeddings of either all word forms or the lemmas, obtained by preprocessing with a lemmatizer. (Right): We use embeddings of all word forms but normalize them in the *embedding space*, integrating naturally into the task model.

times they do not work ideally for specialized vocabularies (e.g. medical language). The creation of such a lemmatizer often requires expert knowledge of the target language.

We propose a novel approach for addressing rich morphology, illustrated in Figure 1. Instead of using a traditional lemmatizer to find the lemmas and using the embeddings for those to represent the content, we do the opposite: We start with the embeddings for all original word forms and then perform lemmatization in the embedding space. This is carried out by a neural network that approximately maps the embeddings of inflected forms into the embeddings of the lemmas. We believe that this may provide embeddings that are better for downstream processing tasks compared to the ones available for the lemmas, for instance when the lemma itself is rare since the model is implicitly able to leverage information across multiple words and cases. Another advantage of lemmatization in the embedding space is easy integration as part of the standard modeling workflow that often builds on neural networks anyway, instead of requiring a separate lemmatizer.

Traditional lemmatization is basically a character-level operation, where grammar rules are used to backtrack the basic form that could have generated the inflected form. We, however, consider word inflections as "bias" in the embedding space, so that the embedding for the inflected word combines (in some unknown way) the semantic meaning of the word and the case information. Consequently, our formulation resembles conceptually the problem of bias removal widely studied in the word embedding literature (Bolukbasi et al., 2016; Brunet et al., 2019). The task in bias removal is to transform the embeddings of individual words such that unwanted systematic biases related to gender etc. disappear. Our approach can be interpreted in this context as a method of removing undesired morphological information while retaining the semantic meaning of the word.

We demonstrate the approach on the Finnish language, restricting the analysis for nouns and adjectives that often contain the most important content words for tasks like document similarity comparison or information retrieval. We use pretrained `fastText` embeddings (Bojanowski et al., 2016) that use subword-level information to provide embeddings for all possible word forms and train a model for mapping them for embeddings of the lemmas using on the dataset extracted from Wiktionary by Durrett and DeNero (2013). The approach is, however, directly applicable to other word classes and languages. Besides the pretrained embeddings, it requires only access to (a) existing list of pairs of lemmas and inflected words as in our case, (b) dictionary and morphological generator, or (c) existing traditional lemmatizer for the language. For instance, `fastText` provides such embeddings for 157 languages, and morphological analyzers or generators exist for most of these.

Besides the core concept of lemmatizing in the embedding space, our main contributions are in the specification of practical details for learning the lemmatizers. We specify four alternative neural network architectures, define a suitable objective function and quality metric, and propose a novel idempotency regularization technique to prevent the models from doing anything else besides the lemmatization. We evaluate the approach in document comparison, outperforming the standard pipeline using traditional lemmatizers, and demonstrate it additionally in the task of word list generation.

An open-source implementation of the method in Python is made available at `https://github.com/jalagus/embedding-level-lemmatization`.

2 Related Work

Even though we are the first to directly consider the task of transforming embeddings to lemmatize words, the general question of addressing rich morphology in distributed representations has been studied from various perspectives.

Cotterell et al. (2018) studied the effect of morphological complexity for task performance over multiple languages. They showed that morphological complexity correlates with poor performance but that lemmatization helps to cope with the complexity. Kutuzov and Kuzmenko (2019) showed a similar effect to hold even with more complex language models, at least for the Russian language. Ebert et al. (2016), in turn, showed that for MRLs we can improve word similarity comparisons by learning Word2Vec embeddings from a lemmatized corpus, rather than training them on all data and lemmatizing while learning the task model.

Kondratyuk et al. (2018) studied supervised lemmatization and morphological tagging using bidirectional RNNs with character and word-level embeddings in MRLs. They showed that a combination of lemma information and morphological tags improve lemmatization and tagging, but may hurt for English. Along similar lines, Rosa and Žabokrtskỳ (2019) suggested using word-embedding clustering to improve lemmatization.

As we consider lemmatization from the perspective of bias removal, our work relates to methods for removal of gender bias (Bolukbasi et al., 2016; Zhao et al., 2017). In this line of work, the embedding space is assumed to encode gender information in specific dimensions, so that bias can be minimized by removing them. The main difference to our work is that their goal is primarily in removing the bias, whereas we look for embeddings that retain the semantic meaning of the word well and that are good for downstream task performance.

3 Evaluation of Lemmatization in Embedding Spaces

A traditional lemmatizer either returns the true lemma or not, but when operating in the embedding space of continuous vector representations the question of correctness needs more attention. We start by discussing the evaluation before proceeding to explain the approach itself that builds on these insights.

First of all, we note that we can use task performance in any downstream task to evaluate the quality – our ultimate goal is in solving the task well, not in learning the embeddings. We will demonstrate this later in the task of document similarity comparison. However, it is highly useful to also have a generic task-independent metric directly measuring the lemmatization accuracy, which can also be used for motivating the objective for training. We want a good word embedding space lemmatizer $M(e_w)$ to simultaneously satisfy two different criteria:

1. Ability to transform any embedding e_w to the embedding of its lemma w', and
2. Ability to retain embeddings of lemmas or lemmatized embeddings as is.

The first criterion is intuitive, matching our goal, but we need to decide how to measure the similarity. For high-dimensional spaces, it is not reasonable to expect a perfect recovery of the embedding $e_{w'}$ itself, but instead, we should count all embeddings that are close enough as correct. To determine 'close enough', we use a simple definition based on neighborhoods: Lemmatization is correct if the *closest neighbor* for the transformed embedding of a word w is the embedding of its lemma w'. We denote by ACC_{LEM} the accuracy of nearest-neighbor (rank-1) retrieval accuracy for w' in the neighborhood of $M(e_w)$, using Euclidean distance for similarity.

The second criterion is imposed as we want to consider the lemmatization step as a black box for which we can feed in arbitrary words, including those that are already lemmas. The lemmatizer should not alter them in any way. We measure this by an indirect metric of ACC_{IDEM}, which corresponds to the rank-1 retrieval accuracy for w' in the neighborhood of $M(M(e_w))$, the output for an embedding e_w passed *twice* through the model. For a more detailed discussion and justification, see Section 4.3.

Together the metrics ACC_{LEM} and ACC_{IDEM} characterize the general ability of any embedding-space lemmatizer in a model-independent way; both are based on retrieval accuracy and can be evaluated without additional assumptions besides the distance measure. We will later use them also to motivate our objective function, a differentiable approximation for their weighted combination.

4 Approach

Denoting an arbitrary inflected form word embedding by e_w and the related lemma word embedding by $e_{w'}$, we wish to learn some mapping $M(\cdot)$ such that $M(e_w|\theta) \approx e_{w'}$. We do this by assuming a parametric model family, a neural network, and learning its parameters θ based on a collection of $(e_w, e_{w'})$ pairs of pretrained embeddings in a supervised fashion. For simplicity of notation, we omit the parameters and simply write $M(e_w)$ instead of $M(e_w|\theta)$ for the rest of the paper.

We hypothesize that inflected forms lie on a specific subspace of the embedding space (see Figure 2) and that we can retrieve the lemmatized forms by a simple, but a possibly nonlinear, transformation in the embedding space. This can be interpreted as the removal of "bias" caused by the inflection. We want this mapping to be lightweight so that it can be integrated as part of a task model with a small computational overhead. Complex transformations are discouraged also because they would increase the risk of altering the semantic content captured by the embedding.

We discuss two alternative ways of lemmatizing in the embedding space. The first approach learns a separate model $M_c(e_w)$ for each word case c so that e.g. partitives and genitives are processed with different models. This allows using simple models even if all of the rich morphology was not constrained in low-dimensional subspaces, and also allows reversing the model for morphological generation (see Section 7).

For the processing of arbitrary words with an unknown case, we can make a function composite of multiple models, so that the output of one model is always fed as input for the next one. For instance, to lemmatize both partitives and genitives we can compute $(M_p \circ M_g)(e_w) = M_{partitive}(M_{genitive}(e_w))$, in either order. Assuming the models do nothing else besides remove the effect of the particular case, then this composite function performs the same operation as either model alone, depending on the case of the input word. We naturally cannot guarantee the transformations work exactly like this, but will later present a regularization technique that specifically encourages the models to focus only on the case removal and show empirically that such function composition of multiple models works well.

The other alternative is learning a single global model $M(e_w)$ that can lemmatize all word forms.

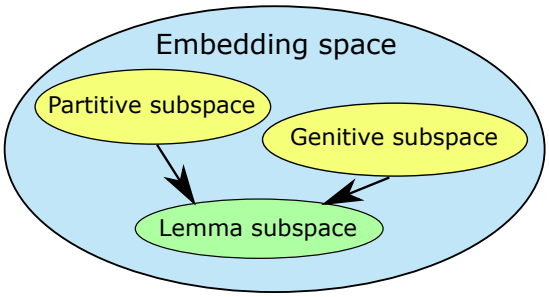

Figure 2: Embedding space as a union of inflected subspaces. Each word class creates a subspace and arrows represents the mappings we wish to learn in order to do lemmatization in the embedding space.

We demonstrate also this approach, but our main focus is on the separate models for each case.

4.1 Neural Architectures

We use feedforward neural networks as models $M_c(e_w)$, restricting the architecture choice for small networks to retain computational efficiency. Both input and output dimensionality needs to match the dimensionality of the embedding, in our case $d = 300$. We investigate empirically four alternative architectures:

1. **Linear**
   $W_1 e_w + b_1$, where $W_1 \in R^{d \times d}$
2. **Simple**
   $W_2 R(W_1 e_w + b_1) + b_2$, where $W_1 \in R^{500 \times d}$, $W_2 \in R^{d \times 500}$
3. **Compression**
   $W_2 R(W_1 e_w + b_1) + b_2$, where $W_1 \in R^{100 \times d}$, $W_2 \in R^{d \times 100}$
4. **Complex**
   $W_3(R(W_2 R(W_1 e_w + b_1) + b_2) + b_3$, where $W_1 \in R^{500 \times d}$, $W_2 \in R^{500 \times 500}$, $W_3 \in R^{d \times 500}$

In all variants, $R(\cdot)$ denotes the rectified linear unit and b_i is a bias term of proper size.

The linear model is motivated by the property of some embeddings encoding various properties as linear relationships (e.g. $king - man + woman \approx queen$) and fast computation. However, there are no guarantees a linear transformation is sufficient for lemmatization and hence we consider also the three simple nonlinear architectures with at most two hidden layers. Other architectures could certainly be used and a more careful choice

of a specific architecture could further improve the lemmatization accuracy, but we will later show that already these lightweight models work well in practice.

4.2 Objective and Training

To learn models such that $M_c(e_w) \approx e_{w'}$ we need to optimize for a loss function that penalizes for difference between $M_c(e_w)$ and $e_{w'}$ for known pairs of w and w'. As explained in Section 3, we will eventually measure the quality by nearest-neighbor retrieval in the embedding space. Directly optimizing for that is difficult, and hence we optimize for a natural proxy instead, minimizing the squared Euclidean distance

$$D(M_c(e_w), e_{w'}) = \|M_c(e_w) - e_{w'}\|^2.$$

Note that often the norm of the embeddings is considered irrelevant and consequently e.g. Word2Vec (Mikolov et al., 2013) used cosine similarity to measure distances. We want to retain the norms that for some embeddings encode information about e.g. word frequency (Schakel and Wilson, 2015) and hence chose the Euclidean distance.

For training the model we need a collection of N pairs of embeddings for words w and their lemmas w'. Assuming an embedding library that provides embeddings for large vocabulary (or even arbitrary word forms, building on subword-level embeddings (Bojanowski et al., 2016)) we simply need some way of constructing these pairs. The two practical alternatives for this are

- Dictionary of lemmas w' and a morphological generator to form w_c for cases c
- Collection of words w and a traditional lemmatizer for obtaining their lemmas w'

For case-specific models we only use pairs corresponding to the case, whereas for the global model we can pool all pairs, potentially having multiple cases for the same lemma in the training data.

4.3 Idempotency Regularization

Any model trained as above learns to map w to w', but we cannot tell what it does for words that are already lemmas or that belong to some other case if training a case-specific model. One could in principle add pairs of (w', w') into the training set to address the former, but to prevent transforming words of other classes we would need similar pairs for *all* possible cases. This would be extremely inefficient.

To avoid transforming the embeddings of other word forms, we propose an alternative of novel regularization strategy encouraging *idempotency*, meaning that the same transformation applied multiple times will not change the output beyond the initial result. We do this by measuring the Euclidean distance $D(M_c(e_w), M_c(M_c(e_w))$ between the output of the model $M_c(e_w)$ (the supposed lemmatized embedding) and the result of passing the input through the model twice, $M_c(M_c(e_w))$. By encouraging this distance to be small we encourage the model to only remove the information about the particular case, without otherwise changing the embedding. Conceptually this is related to regularization techniques like Barone et al. (2017) designed to prevent catastrophic forgetting (Kirkpatrick et al., 2017); both prevent losing the already learned structure while allowing the model to adapt to a new task.

In practice we minimize the objective

$$\begin{aligned} L(e_w, e_{w'}) = \alpha \times D(M_c(e_w), e_{w'}) + \\ (1-\alpha) \times D(M_c(e_w), M_c(M_c(e_w))), \end{aligned} \quad (1)$$

where $\alpha \in [0, 1]$ controls the amount of regularization. With $\alpha = 1$ we only optimize the loss and by decreasing the parameter we start regularizing the solution using idempotency. Note that the extreme of $\alpha = 0$ is not meaningful, since the loss term disappears.

5 Model Validation

We validate the approach and the modeling choices (architecture and regularization), using morphologically rich Finnish as an example language. We first evaluate the performance in a task-agnostic manner, before demonstrating case examples in the following two sections.

Data We validate the approach on Finnish language, using pretrained embeddings provided by the `fastText` library (Bojanowski et al., 2016). The embeddings were trained on Common Crawl and Wikipedia corpora and have dimensionality of $d = 300$.

The lemmatization models are trained on the data provided by Durrett and DeNero (2013) which contains words extracted from the open dictionary *Wiktionary*. It directly provides pairs of

inflected and base forms for words, so we do not need to construct them. For Finnish, the dataset contains 1,136,492 word pairs of adjectives and nouns both in singular and plural, resulting in roughly 42,000 word pairs per word case. Each row in the dataset is a pair of form (w, w') which are then transformed to pairs of word embeddings $(e_w, e_{w'})$ using the `fastText` library.

Training We use `AdamW` optimizer (Kingma and Ba, 2014; Loshchilov and Hutter, 2018) with a learning rate of 0.0002 and a batch size of 32 for training the models in all experiments, but all reasonable stochastic optimization algorithms would work. We separately validated in preliminary tests that running the optimization until convergence of the training objective does not result in overfitting, and hence for the rest of the experiments we used 50 epochs for training to make sure the models are fully converged. In practice, 20-30 epochs were always enough. All experiments shown here are efficient, so that training individual models on consumer-grade 8-core CPU was done in the order of minutes.

Model architectures To compare different architectures, we trained individual models $M_c(e_w)$ on all 15 word cases of Finnish with $\alpha = 1.0$ (i.e. no regularization), not separating plural and singular word cases so that always 10,000 word pairs were used for training and 1,000 for testing. The word pairs for training and test sets were chosen randomly. For the final score, we averaged 10 different runs over randomized splits of the data so that the splits were the same for all models for each run.

Table 1 compares the four different architectures in terms of metrics explained in Section 3, presenting the average accuracy over all word cases (the results are consistent over different cases, not shown here). The main result is that except for the *compression* architecture the accuracies ACC_{LEM} are very similar. This suggests there may not be a specific low-dimensional subspace that is sufficient for lemmatization, but that it can be modeled with fairly simple architectures nevertheless. In terms of ACC_{IDEM}, all models here coincidentally converge to the same value that is close to perfect despite not regularizing for idempotency.

We also trained a global model $M(e_w)$ for lemmatizing all cases using the *simple* model architecture, using a combined data set of 50,000 examples covering the different cases and 5,000 word pairs for evaluation. Note, however, that the evaluation set was not the same as for the case-specific models that all used only pairs for the specific case. Hence the numbers in Table 1 are not directly comparable, but we can still confirm that also the global model learns to lemmatize well.

Model	$\mathrm{ACC_{LEM}}$	$\mathrm{ACC_{IDEM}}$	Time/epoch (s)
linear	0.908	0.978	1.363
compression	0.870	0.978	1.807
simple	**0.915**	0.978	3.044
complex	0.911	0.978	3.669
global	0.974	0.998	11.145

Table 1: Lemmatization accuracy (ACC_{LEM}) and idempotency criterion (ACC_{IDEM}) for alternative network architectures for case-specific models, averaged over all 15 word cases. The global model can process all cases, but the numerical accuracy is not directly comparable due to a different number of test instances.

Function composition and idempotency regularization When training separate models for each word case c, we need a function composition of multiple models in order to process arbitrary input word forms. To perform this, we need idempotency regularization to prevent individual models from transforming words of wrong cases.

Table 2 demonstrates the effect of the regularization parameter α for an example sentence, using two models trained for lemmatizing genitives and partitives and their combination as function composition. For very small α already the individual models fail due to almost ignoring the main task, whereas for very large α (no regularization) the composition breaks. With $\alpha = 0.4$ we can accurately lemmatize both forms.

6 Application: Document Comparison

To demonstrate the method in a typical application, we consider the task of document comparison where the lemmas of content words often provide sufficient information on similarity. We use a dataset provided by the Finnish national broadcasting company *Yle*[1] containing news articles written in easy-to-read Finnish. We created an artificial dataset by splitting news articles into

[1] http://urn.fi/urn:nbn:fi:lb-2019121205

α	Word case	Example sentence											
-	original	Leijona	oli	saanut	paitsi	hyvän	ja	nöyrän	mielen	myös	monta	uutta	ystävää
0.1	genitive	Leijona	oli	saanut	näinen	hyvä	pipopää	nöyrä	tahdonvoima	näinen	iso	uusi	ystävää
0.1	partitive	Leijona	oli	saanut	paitsi	hyvä	pehmyt	nöyrä	mielen	myös	muutama	uusi	ystävä
0.1	gen + part	Leijona	oli	syynännyt	näinen	hyvä	pipopää	nöyrä	tunteellisuus	näinen	pieni	uusi	tyttökaveri
0.4	genitive	Leijona	oli	saanut	paitsi	hyvä	ja	nöyrä	mielen	myös	monta	uutta	ystävää
0.4	partitive	Leijona	oli	saanut	paitsi	hyvän	ja	nöyrän	mielen	myös	monta	uusi	ystävä
0.4	gen + part	Leijona	oli	saanut	paitsi	hyvä	ja	nöyrä	mielen	myös	monta	uusi	ystävä
0.7	genitive	Leijona	oli	saanut	paitsi	hyvä	ja	nöyrä	mielen	myös	monta	uutta	ystävää
0.7	partitive	Leijona	oli	saanut	paitsi	hyvän	ja	nöyrän	mielen	myös	useampi	uusi	ystävä
0.7	gen + part	Leijona	oli	saanut	paitsi	hyvä	ja	nöyrä	mielen	myös	monta	uusi	ystävä
1.0	genitive	Leijona	oli	saanut	paitsi	hyvä	ja	nöyrä	mielen	myös	monta	uutta	ystävää
1.0	partitive	Leijona	oli	saanut	paitsi	hyvän	ja	nöyrän	mielen	myös	monta	uusi	ystävä
1.0	gen + part	Leijona	oli	saanut	paitsi	hyvä	yskäkin	nöyrä	tahdonvoima	myös	muutama	uusi	ystävä
0.3	global	Leijona	oli	saanut	paitsi	hyvä	ja	nöyrä	mielen	myös	monta	uusi	ystävä
-	ground truth	Leijona	oli	saanut	paitsi	hyvä	ja	nöyrä	mieli	myös	monta	uusi	ystävä

Table 2: Idempotency regularization for function composition of separate models for lemmatizing genitives and partitives. Both too large and small α introduce mistakes for this example sentence, but with $\alpha = 0.4$ and the alternative of global model the result is near perfect. The words in genitive form in the original sentence are $\{hyvän, nöyrän, mielen\}$, and the words in partitive form are $\{uutta, ystävää\}$

two halves and try to predict which two parts belong together by ranking the articles via average vector document representations. We take only a subset of the data, using the first 10,000 news articles from the first three months of the year 2018.

We compare the proposed approach against a conventional pipeline that first lemmatizes the words using the uralicNLP library (Hämäläinen, 2019) (and then uses embeddings for the lemmas for the task) and a pipeline that directly uses the embeddings for all word forms. For the proposed approach we perform lemmatization in the embedding space for four different cases and their combinations, using the *simple* architecture.

For all methods, we form a representation for the document by computing the mean of the word embeddings for all words in the document and use cosine similarity between these mean embeddings to compare documents. One could alternatively consider richer document representations (Wieting et al., 2015; Arora et al., 2017; Gupta et al., 2020) or more accurate similarity metrics (Torki, 2018; Lagus et al., 2019) that might improve the overall accuracy, but we chose the most commonly used approach that is easy to understand to focus on demonstrating the effect of the lemmatization.

We measure performance by retrieval accuracy, by computing the rank of the second half of a given document amongst the set of all 10,000 second halves. Figure 3 shows the overall performance of the different model variants as a function of the regularization parameter, measured by rank-1 accuracy. We observe three clear results: (a) all ways of lemmatization clearly improve the task performance compared to no lemmatization at all, (b) lemmatization in the embedding space using case-specific models is considerably better than the alternatives of traditional lemmatizer and the global model lemmatizing in the embedding space, and (c) idempotency regularization is crucial, but the method is extremely robust with respect to the specific choice of α – all values between 0.2 and 0.9 result in almost identical performance.

Table 3 illustrates the task performance in more detail for models trained using good choices for the regularization parameter α, measured using retrieval accuracy with different ranks. The results are consistent over the ranks, with case-specific lemmatizers in the embedding space consistently outperforming the other methods.

7 Application: Word List Generation

Even though our main goal is to learn lemmatizers, we note that that the approach is more general. Instead of training a lemmatizer $M_c(e_w) \approx e_{w'}$, we can use the exact same architectures and data for training $G_c(e_{w'}) \approx e_w$ to learn *generators* that provide the embedding for the inflected form for some particular case c.

We demonstrate this via the simple application of word list expansion, which could be used simi-

Model	Word case	α	R@1	R@2	R@3	R@4	R@5	R@6	R@7	R@8	R@9	R@10
simple	gen	0.8	0.368	0.455	0.502	0.534	0.560	0.582	0.599	0.613	0.625	0.636
simple	gen + part	0.8	0.390	0.483	0.533	0.569	0.594	0.613	0.632	0.647	0.658	0.669
simple	gen + ine + part	0.5	0.395	0.491	0.540	0.573	0.597	0.620	0.636	0.649	0.663	0.674
simple	gen + ine + ela + part	0.5	0.390	0.482	0.533	0.567	0.594	0.614	0.631	0.645	0.657	0.668
global	-	0.9	0.329	0.411	0.458	0.488	0.513	0.532	0.548	0.561	0.573	0.585
lemmatizer	-	-	0.311	0.391	0.434	0.464	0.485	0.503	0.518	0.532	0.543	0.552
none	-	-	0.286	0.362	0.404	0.431	0.454	0.474	0.490	0.501	0.514	0.526

Table 3: The best combinations of each model version averaged over 10 different subsets of the news data. R@K means that we rank the documents by similarity and measure the accuracy of the relevant document being within the top K documents.

larly to query expansion for retrieval tasks. Given a list of words w' provided in base form and their embeddings $e_{w'}$, we form a list of embeddings for different inflected forms. We trained case-specific models $G_c(e_{w'})$ similar to before with different values for α, observing a similar trend: the method is robust for the choice, as long as extreme values are avoided.

Table 4 illustrates the method for the word list $\{jääkiekko, Suomi, Venäjä\}$ ($\{ice\,hockey, Finland, Russia\}$ in English) one could use as keywords for searching information about ice hockey matches between the two countries. We show here the words with the embeddings closest to the ones provided by the generator models to verify it works as intended, but in real use, we would naturally use the transformed embeddings directly for the retrieval task – they are likely to be better representations especially for rare cases for which the actual pre-computed embedding e_w is likely to be noisy.

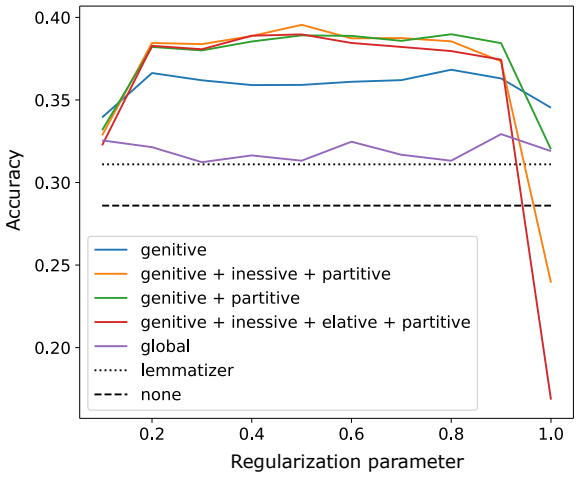

Figure 3: The effect of the regularization parameter (the α parameter) on full-length document comparison task using rank-1 accuracy as the scoring method. There is a notable improvement over the baselines (lemmatizer and none) when using our models with idempotency regularization parameter chosen within the range $(0.2, 0.9)$, and the improvement is highly insensitive to the specific value of the parameter.

8 Conclusions

For MRLs lemmatization helps in many tasks. We showed that the conventional pipeline using traditional lemmatizers as preprocessing can be replaced by lemmatization in the embedding space. Already simple neural networks can transform the embeddings of inflected words so that the closest word in the embedding space is of the correct lemma. This verifies lemmatization in the embedding space is possible, but in real applications, we naturally would not convert the result back to the lemma. Instead, any downstream task simply processes the lemmatized embeddings directly.

We showed that the method outperforms conventional lemmatization preprocessing in the document similarity comparison task, which implies we are not merely learning to replicate the exact lemmatization but instead learn embeddings that better capture the word content. We hypothesize this is related to how rare words are represented in the embedding space; for rare words, the embeddings for all word forms are unreliable, including the one for the lemma itself. Subword-level embeddings, like `fastText` used in our experiments, may still be able to learn sensible embeddings for the collection of all inflected forms together, and by lemmatizing in the embedding space we borrow some information from all of the forms. In other words, we argue that the approximate lemmatization performed by the neural network may have the regularizing ability to reduce noise in embeddings of rare words so that the 'approximation' is actually better than the target embedding used during training.

Word case	Expanded form	Ground truth
genitive	jääkiekon	jääkiekon
genitive	Suomen	Suomen
genitive	Venäjän	Venäjän
inessive	*jääkiekkossa*	jääkiekossa
inessive	Suomessa	Suomessa
inessive	Venäjässä	Venäjässä
elative	jääkiekosta	jääkiekosta
elative	Suomesta	Suomesta
elative	Venäjästä	Venäjästä
partitive	jääkiekkoa	jääkiekkoa
partitive	Suomea	Suomea
partitive	Venäjää	Venäjää
illative	jääkiekkoon	jääkiekkoon
illative	Suomeen	Suomeen
illative	Venäjälle	Venäjälle

Table 4: Example word list expansion generated for the word list $\{jääkiekko, Suomi, Venäjä\}$ ($\{ice\,hockey, Finland, Russia\}$) using morphological generator models for genitive, inessive, elative, partitive, and illative cases with regularization parameter $\alpha = 0.4$. Note the mistake for the inessive case of "jääkiekko", which should be "jääkiekossa" and not "jääkiekkossa" – the word has the correct "-ssa" suffix but the root is incorrect. It is also worth noting that "jääkiekkossa" is not a valid word form in Finnish at all, but the `fastText` library provides embeddings for arbitrary strings using sub-word information. The embeddings for the two forms are likely very close, and hence the mistake would have no effect in retrieval tasks.

In this work we presented the overall concept for lemmatization in the embedding space and experimented on various technical choices, building the basis for future development. Our main findings were that a global model can perform lemmatization well when measured only by accuracy, but for the task of document comparison, we reached considerably better results by function composition of case-specific models. To make this possible we proposed a novel idempotency regularization, and showed that the approach is highly robust for the choice of the regularization parameter, making it essentially parameter-free. Finally, we note that even though we demonstrated the approach for an example MRL language Finnish and only for lemmatization of nouns and adjectives, the method is general and directly applicable for other languages and word classes.

Acknowledgements

This work was supported by the Academy of Finland Flagship programme: Finnish Center for Artificial Intelligence, FCAI.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *Proceedings of International Conference on Learning Representations*.

Antonio Valerio Miceli Barone, Barry Haddow, Ulrich Germann, and Rico Sennrich. 2017. Regularization techniques for fine-tuning in neural machine translation. *arXiv preprint arXiv:1707.09920*.

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

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *Advances in neural information processing systems*, 29:4349–4357.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Marc-Etienne Brunet, Colleen Alkalay-Houlihan, Ashton Anderson, and Richard Zemel. 2019. Understanding the origins of bias in word embeddings. In *International Conference on Machine Learning*, pages 803–811. PMLR.

Ryan Cotterell, Sabrina J. Mielke, Jason Eisner, and Brian Roark. 2018. Are all languages equally hard to language-model? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 536–541, New Orleans, Louisiana. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of the 2013 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1185–1195, Atlanta, Georgia. Association for Computational Linguistics.

Sebastian Ebert, Thomas Müller, and Hinrich Schütze. 2016. Lamb: A good shepherd of morphologically rich languages. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 742–752.

Vivek Gupta, Ankit Saw, Pegah Nokhiz, Praneeth Netrapalli, Piyush Rai, and Partha Talukdar. 2020. P-sif: Document embeddings using partition averaging. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7863–7870.

Mika Hämäläinen. 2019. UralicNLP: An NLP library for Uralic languages. *Journal of Open Source Software*, 4(37):1345.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Daniel Kondratyuk, Tomáš Gavenčiak, Milan Straka, and Jan Hajič. 2018. LemmaTag: Jointly tagging and lemmatizing for morphologically rich languages with BRNNs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4921–4928, Brussels, Belgium. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71.

Andrey Kutuzov and Elizaveta Kuzmenko. 2019. To lemmatize or not to lemmatize: How word normalisation affects ELMo performance in word sense disambiguation. In *Proceedings of the First NLPL Workshop on Deep Learning for Natural Language Processing*, pages 22–28, Turku, Finland. Linköping University Electronic Press.

Jarkko Lagus, Janne Sinkkonen, Arto Klami, et al. 2019. Low-rank approximations of second-order document representations. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*. ACL.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

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.

Rudolf Rosa and Zdeněk Žabokrtskỳ. 2019. Unsupervised lemmatization as embeddings-based word clustering. *arXiv preprint arXiv:1908.08528*.

Adriaan MJ Schakel and Benjamin J Wilson. 2015. Measuring word significance using distributed representations of words. *arXiv preprint arXiv:1508.02297*.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152. IEEE.

Marwan Torki. 2018. A document descriptor using covariance of word vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 527–532.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2017. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2979–2989, Copenhagen, Denmark. Association for Computational Linguistics.