Journal of Theoretical and Applied Information Technology
15th January 2022. Vol.100. No 1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ENHANCING DOCUMENT REPRESENTATIONS WITH SYNONYMS GRAPH NODE EMBEDDINGS

[1]ROMAN SHAPTALA, [2]GENNADIY KYSELOV

[1]PdD Student, NTUU "Igor Sikorsky KPI", Institute for Applied System Analysis, Department of System Design, Kyiv, Ukraine

[2]Associate Professor, NTUU "Igor Sikorsky KPI", Institute for Applied System Analysis, Department of System Design, Kyiv, Ukraine

E-mail: [1]r.shaptala@gmail.com, [2]g.kyselov@gmail.com

ABSTRACT

Document representations are a key element in solving natural language processing problems, including document classification, clustering, and topic modelling. Modern deep learning-based techniques are able to build such representations with great success. However, they require a lot of data to do so, thus are not applicable in the low-resource setting. Recently, low-resource NLP methods focused on transfer learning approaches which assume the existence of pretrained models, as well as data augmentation approaches which expand available datasets. Our key idea is to use a synonyms dictionary as a transfer learning tool instead of a source for token-level document augmentation. Consequently, the objective of this study is to investigate the improvement of existing document embedding methods in a low-resource setting by the incorporation of synonym dictionary information. Our main research contributions are the demonstration of the effectiveness of synonym node embeddings as a transfer learning method for building document representations and the proposal of a method for enhancing document representations by mixing them with an average of node vectors of words from a synonyms graph of the low-resource language. The experiments show a 2% F-score improvement for Kyiv City petitions topic classification in Ukrainian. The analysis and optimal hyperparameters for training a Node2Vec graph embeddings model as well as the weighted sum fusion of baseline and synonym embeddings are provided. Future work can explore the impact of other node embedding methods or the application of other linguistic dictionaries in a similar fashion.

Keywords: *Low-Resource Natural Language Processing, Node Embeddings, Document Embeddings, Synonyms, Document Classification*

1. INTRODUCTION

Linguistic analyses that operate on the semantic and pragmatic level are common natural language processing tasks. These are the problems that require the machine to understand the meaning of sentences or documents, e.g., text summarization, sentiment analysis, and topic clustering. Usually the general term of "document" refers to any semantically coherent piece of text – a sequence of words – it can be a sentence, a paragraph, or a book. It is not difficult to imagine the range of applications that benefit from automated document modelling, be it customer review categorization or disease diagnosis.

Since natural languages are not "native" to computer processors, the search of a way to optimally represent text digitally is an active area of research. The general concept of such methods is to encode texts as dense vectors of floats in an embedding space that preserves linguistic properties of the objects in question. This turns freeform text into mathematical entities and makes it easy to compare documents by distance functions as well as cluster them.

Currently however, the most sophisticated methods for building document representations are training them as a part of an end-to-end deep learning process via highly-parameterized models like transformers [1]. Although, extremely powerful, the method requires huge amounts of data and processing time. Our research focuses on how to build document embeddings in a low-resource setting, where trainable corpora is limited, e.g., in an underrepresented language on the Web.

Most of the natural language processing (NLP) research nowadays focuses on big data applications or explores the most popular languages,

English in particular. As a result, a lot of approaches developed assume the availability of significantly sized resources from which a corpus can be retrieved. For example, pre-training of BERT [1] is done on the BooksCorpus [2] and English Wikipedia which results in 3,300M words combined. Such huge bodies of text are not available for other languages, even with today's prevalence of user-generated content on the Internet.

Low-resource scenarios can have different roots, but one thing that they have in common is the limited amount of labeled data. In NLP typically there are several possible reasons for this problem: a language that is unpopular and quality labelling is costly and impractical, or a task that is non-standard and heavily relies on domain-specific linguistic structures. Since there is only a limited amount of information in such datasets, most of the solutions in low-resource settings try to get as much additional bits of information from other sources as possible. This leads to the expansion of assumptions that are needed to be satisfied before the algorithms can be used. As a result, the development of new methods with fewer or less restrictive assumptions is an important natural language processing task.

The overall objective of this study is to investigate the improvement of existing baseline document embedding methods in a low-resource setting by the incorporation of synonym dictionary information. This entails the development of synonym words and synonym document representations and the analysis of methods of their combination with baseline document representations. Furthermore, this paper aims to incite a discussion on the fusion of natural language processing methods and dictionary-based information via graph embedding methods.

2. LITERATURE REVIEW

There exist many ways to build document encodings. The most basic one is Bag of Words (BOW) [3], where each word is represented as an index in a dictionary and the document is just a sparse vector of indices of its words. A popular generalization of this method, called Term Frequency-Inverse Document Frequency (TF-IDF) [4], addresses BOW's issue of ignoring the overall word-document frequency statistics by reweighting word contributions to the document vector based on the fraction of documents that include the word. This generally makes common words that do not contain much information to be less impactful on the resulting representation. These two methods are examples of statistical representations. They do not require any training and work well in a number of applications. However, resulting document vectors are sparse and very high-dimensional, as their size is defined by the size of the dictionary. This makes them difficult to query or use in machine learning pipelines.

An alternative branch of methods is based on the combination of word embeddings built on top of the distributional hypothesis: words that appear in similar contexts have similar meanings [3]. These word vectors are usually trained as a part of a self-supervised process or a different machine learning pipeline while solving an NLP task end-to-end. The resulting word embeddings are dense and low-dimensional, which makes them more appealing for transfer learning [5]. This type of methods is also called neural, because they were first introduced as a part of neural network language model [6]. However, it is unclear how to combine word embeddings into a document embedding. Several methods have been proposed for this, the most popular baseline being – take a per-dimension average of word vectors present in the document [7]. Namely, given a set of word vectors S, where $S_i \in R^m \ \forall i \in [z]$ and z is the size of word embedding dictionary, the r'th element of the baseline embedding of document j consisting of n words becomes:

$$W_{jr} = \sum_{i=1}^{n} S_{ir} / n \qquad (1)$$

A recent survey of low-resource natural language processing methods was done by Michael A. Hedderich et. al in [8]. Most of the approaches in this setting fall into two categories: generating additional labeled data or transfer learning. The categorization of methods to solve NLP problems in a low-resource setting is shown in figure 1.

Generation of substitutes for gold-standard data is an idea that came from computer vision [9], where the transformations of images, e.g., cropping, rotation or zooming, create additional training points and enhance the performance on a wide-variety of

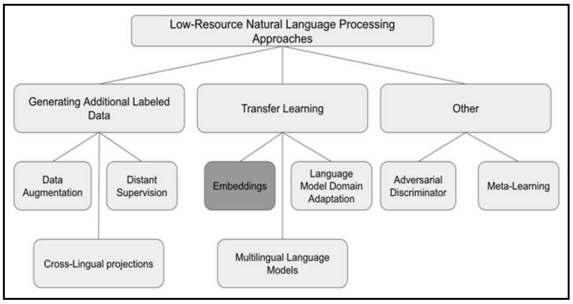

Figure 1: Classification of low-resource natural language processing methods

tasks. In natural language processing, however, the transformations are less obvious and require extensive knowledge of the language in question. Moreover, they can introduce wrong or unnatural examples into the dataset, so have to be used with care. These data augmentation techniques include: the simulation of keyboard error (substituting words that are frequently mistyped), synonym and antonym substitution, random word splitting etc. A different way of generating additional labeled data is distant supervision [10]. It is mostly concerned about labeling unlabeled data based on simple rules, entities in knowledge bases or ontologies. Distant supervision works well for a number of low-resource settings because it is able to extract high-precision patterns that uniquely identify certain classes. This approach is generally used on named-entity-recognition or various syntactic and morphological tasks, and is rarely applicable to the problem we are trying to solve. Moreover, it assumes that unlabeled data exist which is not always possible in a low-resource setting. For example, authors of [11] propose to generate weakly labeled data to combat the scarcity of available resources via contextualized representations for instances where rules cannot be used. This is done by building a separate self-trained student model which suggests pseudo-labels for datapoints where typical rule-based weak supervision does not work. While the approach leads to significant improvements in accuracy for question and spam classification, the method relies on the existence of additional data points which can be labeled via weak supervision. Depending on the source of the lack of resources, for certain settings distant supervision methods might not be available. For example, highly specific or recently designed engineering entities might not exist in available knowledge bases, so it is impossible to build rules for distant supervision that cover them.

Another interesting data augmentation approach that has been introduced recently is data augmentation using pre-trained transformer models [12]. Here the effectiveness of using pre-trained language models for new training samples generation is explored. The authors show that autoencoding, autoregressive and seq2seq pre-trained models can be conditioned on available labels to generate new labeled datapoints thus enhancing the performance of typical document classification models. Although this method is extremely powerful given the strength of modern transformer-based pre-trained models, it still assumes that these models exist and were trained on a huge corpus of natural language data. This makes it not applicable to any low-resource languages which lack labeled or unlabeled data to train such transformers.

Transfer learning methods, on the other hand, are less concerned about data and more concerned about prebuilt models that capture language features and can be reused between tasks. This type of approaches is highly successful because once the original model is trained, its weights are easily sharable between tasks and might contribute positively and add out-of-scope knowledge to a new task. Still, however, these transferred models need to be learned, which requires a separate dataset, usually large and unlabeled, and a significant time and hardware investment for their training process. Moreover, for transfer learning to work, the original problem should be similar to the one at hand, so that the transfer is positive. This can be difficult to achieve when you're trying to solve a domain-specific problem.

A comparison of basic statistical and transfer learning methods for publications clustering was done in [13]. By testing TF-IDF, Word2Vec averaging and end-to-end methods on a dataset of 15907 documents the authors conclude that there was little difference between their performance, but Word2Vec averages are the optimal strategy towards document clustering because of its contextual sensitivity. One limitation of their research is that they've only tested one clustering algorithm, namely K-Means clustering, which requires the number of clusters to be determined in advance, as well as the use of Euclidean distance. Also, this clustering method assumes that clusters have normal distributions in the embedding spaces, which may not be the case in all of the domains.

To our knowledge, using synonyms graph information as a transfer learning method for document representation has not been explored yet. This is contrary to the common data augmentation method that builds additional data samples by swapping synonymous words in documents [14]. This approach is not easily transferable and requires custom logic to work. Having an embedding-based transfer learning solution would facilitate decoupling of language synonymy representation and the process of document classification models training, so that basic word relatedness information could be shared between tasks.

3. PROPOSED SOLUTION

The method we introduce in this paper can be classified into the transfer learning category. Most of the transfer learning ideas rely on an abundance of unlabeled texts in the same low-resource language,

for example the Wikipedia [15], or a reuse of labeled resources in other languages via cross-lingual models. Our assumption is different – we do not expect to get additional implicit labeled data, but we do want to leverage extra language information that is provided by a synonym dictionary, thus it needs to exist in the task-related language. We believe that this assumption is true in a lot of low-resource languages, as synonymy is a fundamental linguistic feature that has been studied extensively for a long time [16] and the dictionaries are systemically updated by linguists in different countries. A synonym dictionary is a mapping from words to lists of their synonyms.

A typical document classification pipeline consists of building document vectors by any of the aforementioned methods, e.g., Word2Vec [5] word vectors averaging, then training a supervised learning model to classify the document. We propose to add another source of document vectors, namely synonym vectors, to the baseline document vectors, and proceed with learning on the mixed set of embeddings. The intuition behind this is that by adding explicit synonymy information the classifier will be able to connect documents that were previously seen as lacking common elements, thus strengthening its decision making.

Three key questions arrive when we talk about adding synonym dictionary information to already established document classification methods:

1. How do we represent synonymy and convert the dictionary into a format that is useful?
2. How do we combine synonyms representations into document synonyms representations?
3. How do we combine document synonyms representations with baseline document vectors?

3.1 Synonyms Graph Node Embeddings

Since the most natural mathematical representation for synonymy is a graph where nodes represent words and edges represent synonymy, we propose to use graph embedding techniques [17] to encode its nodes. These techniques represent networks in a vector space, preserving the key features of the graph. This allows for an additional layer of abstraction for a multitude of graph-related tasks, as node and edge embeddings can be transferred or fit a typical classifier on top of, instead of building a non-generalizable solution to a concrete graph.

Let G denote the graph of synonyms in a certain language. Every word that has a synonym forms a set of nodes V and if two words i and j are synonymic, there exists an edge from the set of edges E between V_i and V_j. The total number of nodes n is equivalent to the number of unique words in the synonyms dictionary. We do not put weights on edges, because of the lack of information, but future research may look at adding weights proportional to the frequency of these words appearing in the same contexts in a large corpus of texts. Another weighting scheme that might be applicable is when partial synonyms (those that have subtle differences) get lower weights than absolute synonyms. The graph is also undirected, because of the underlying symmetry of synonymy. Then, a node embedding is a mapping $f: V_i \rightarrow H_i \in R^d \ \forall i \in [n]; d << n$ and the mapping keeps graph structural and connectivity properties intact.

A great review of modern graph embedding approaches was written by Palash Goyal and Emilio Ferrara [18]. There, the classification, advantages and disadvantages of aforementioned methods are listed. We summarize the classification in figure 2 and choose a random walk-based method, called Node2Vec [19], as the way to create synonyms graph node embeddings in our experiments. Node2Vec is trained to maximize the likelihood of preserving network neighborhoods of nodes. The process basically consists of two steps: biased random walk generation and node weights optimization. The first step outputs sequences of synonyms with different length, so that the model can learn both local and global features of the synonyms graph. The second step runs an iterative procedure that updates randomly initialized node embeddings according to the embeddings of its context nodes. Context nodes are the ones on the left and right of a certain node in a sequence generated by the random walk. The choice was made based on the fact that Node2Vec models community structures and structural equivalence between nodes well [20]. This also makes our approach an example of the embedding subcategory of transfer learning methods.

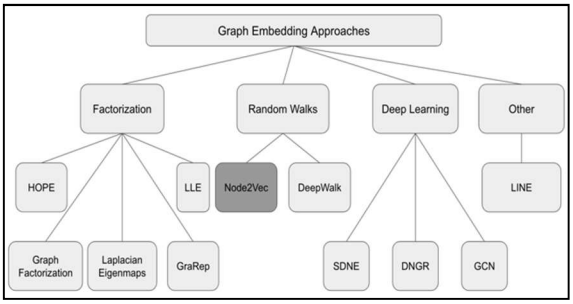

Figure 2: Classification of graph embedding approaches

To create a synonym embedding for the whole document, we need to aggregate node embeddings corresponding to the words appearing in the document. A procedure, similar to how baseline Word2Vec aggregation for documents works (Equation 1), can be used. A document of *t* words produces a set of embeddings, each of size *d*. These vectors get averaged along each dimension to produce a single vector of size *d*. If a word is not present in the synonym dictionary it is ignored and do not take part in any of the calculations related to synonym embeddings. If every word in a document does not have any synonyms, so that it's not in the dictionary, synonym embedding becomes a vector of zeros of size *d*. This is done, so that such documents do not change their representation during the combination with baseline document embeddings if summation is used.

3.2 Embeddings Fusion

As for the combination of synonym with document representations, we can apply approaches explored in multimodal machine learning studies. In this context the process is called fusion, and the least complex form of fusion is Simple operation-based fusion [21]. It can be of two types: concatenation and weighted sums. Figures 3 and 4 show how both of these methods can be applied for the solution of the given task.

Both methods perform a transformation g: $W_i \in R^m$, $H_i \in R^d \rightarrow U_i \in R^k \ \forall\, i \in n$; where W_i is a document vector of size *m* obtained by a classical method, H_i is a synonyms vector of size *d*, and *k* is the resulting size of the embedding. For concatenation $k = m + d$. Weighted sums require *m* to equal *d* and the resulting size *k* is equal to *m*.

There is an important detail in the weighted sums scheme of fusion – the weights that baseline and synonyms vectors are multiplied by. More

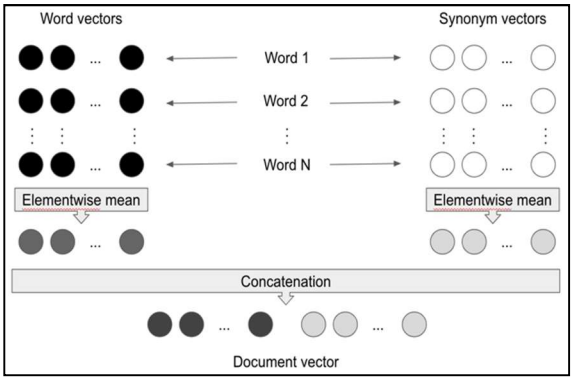

Figure 3: Concatenation-based document and synonym vectors fusion

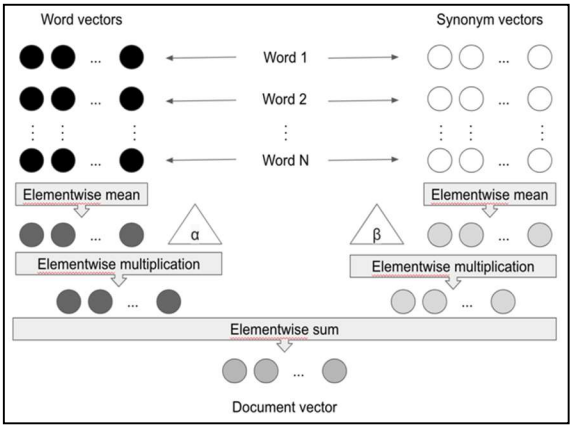

Figure 4: Weighted summation-based document and synonym embeddings fusion

specifically, the equation that governs weighted sum fusion of baseline and synonym embeddings in this work is:

$$U_i = \alpha W_i + \beta H_i \qquad (2),$$

where W_i is the baseline document vector, H_i is the synonym document vector, α is the weight of baseline embeddings and β is the weight of synonym embeddings.

Both hyperparameters α and β can be interpreted as the level of contribution of each of the types of embeddings towards the final vector. We do not restrict β to be $1-\alpha$, with an intuition that synonym vectors add new information to the task, and should not get in the way of baseline vectors if it is not the optimal way to do.

Since statistical encoding approaches like BOW or TF-IDF produce sparse and highly dimensional vectors, the fusion step for them is unclear. One might argue that it is possible to use dimensionality reduction techniques to convert these vectors into low-dimensional dense representations. However, this would make the experiment more convoluted by the introduction of a separate set of meta- and hyperparameters. This is why we restrict the selection of baseline document embeddings to Word2Vec-based and explore the impact of both fusion approaches on the document classification task in a low-resource setting. However, we compare the performance of TF-IDF-based classification method with the proposed one.

All in all, main contributions of this paper are two-fold: (1) we demonstrate the effectiveness of synonym node embeddings as a transfer learning method for building document representations; (2) we introduce a new method to improve document representation and classification in a low-resource setting through the creation, fusion, and use of

synonym embeddings trained on a graph of synonyms via node embedding techniques. This shows that synonym graphs can be used not only as a data augmentation technique for new samples generation, but also as a document representation improvement method. We also provide practical recommendations on the implementation of different parts of the proposed method, including word, document, and synonym dictionary embedding model hyperparameters, document preprocessing steps and the choice of an embeddings fusion algorithm.

4. EXPERIMENTAL SETUP

For the experimental part of our research, a dataset of Kyiv City petitions was used. It has been collected automatically from the Kyiv State Administration Petitions portal [22] which gives an opportunity for citizens to influence urban decision making. All of the petitions on the website are manually classified into different categories: 'Transport', 'Landscaping and environment', 'Streets naming', 'Housing', 'Urban planning', 'Culture and education', 'Health and sports', 'Parking', 'Spontaneous trade', 'Social protection', 'Animals', 'Law and order', 'Budget', 'Waste' and 'Other'. Automation of the process of petition categorization could save administration workers some time as well as make the pipeline of petitions analysis more manageable. This task is an example of a low-resource setting because of two reasons: the low amount of labeled data and the Ukrainian language the petitions are written in. We describe the dataset in more detail in the next section. As for the language, although there are estimated 40 million native speakers, it is very underrepresented on the Web - only 0.6% of websites have contents written in Ukrainian [23] - so there is a limited number of high-quality resources which can be used to train or enhance models. This is why the search of alternative techniques is a very practical and active area of research.

4.1 Classification Dataset

Originally, the dataset of Kyiv City Petitions consists of 6560 petitions. Since most of Kyiv citizens can write in at least two languages, Ukrainian and Russian, language classification had to be done on every point in the dataset. We used pretrained FastText language classifier [24], [25] - a neural network-based model - to automatically label all of the petitions with their languages. Our previous work explored the effectiveness of this method and it turned out to be very reliable with almost perfect 0.99 precision, recall and F1-scores. After the removal of non-Ukrainian petitions, as well as the process of deduplication, only 5332 are left for the remaining research.

One common problem in all low-resource classification settings is the imbalance of target labels. Kyiv City Petitions dataset is no exception and this imbalance is at the core of the classification task as different aspects of living in an urban environment produce different amount of citizen outcry. Some of the classes in our dataset take less than 1% of all samples each, so we decided to remove them from the investigation until they collect enough mass. This further reduced the number of datapoints available for training to 4993. We show the final distribution of datapoints between classes in figure 5.

Length of the sequence of words is an important consideration for the selection of methods to model it, but for the sake of our research, a document is a varied-length text of several paragraphs. The distribution of the number of tokens throughout documents is shown in figure 6. Although most of the documents in the Kyiv City petitions dataset have a relatively low number of words, there are also outliers with as low as a single word and as high as 860 words. Other statistical features of the dataset include: total number of words – 368046, number of unique words – 65374, average words per document – 74, average length of a document – 780 characters, average characters per word – 7.

In order to increase the performance of machine learning methods, the dataset is then put

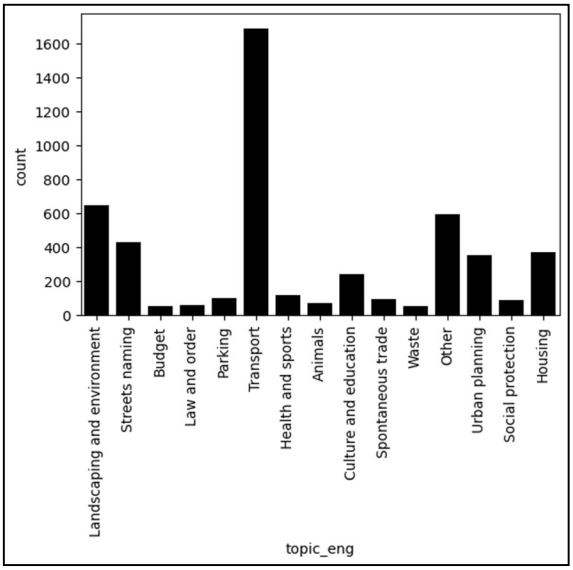

Figure 5: Kyiv City petitions dataset document classification label distribution

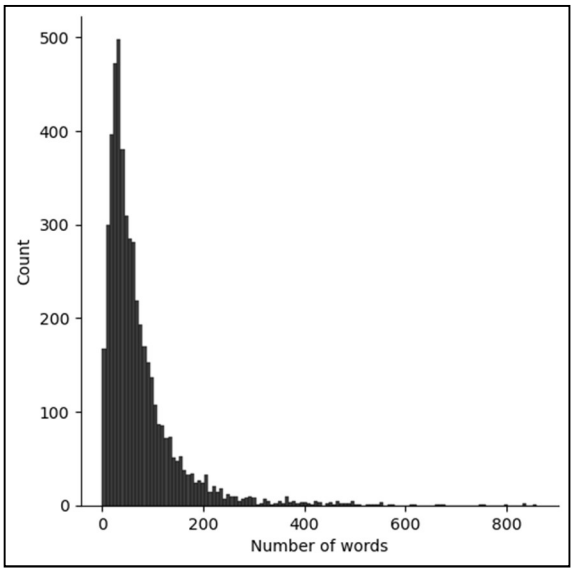

Figure 6: Kyiv City petitions dataset documents word count distribution

through a number of preprocessing steps. The first one is the removal of stop-words – frequent but mostly irrelevant to the classification task tokens with an intention of limiting their impact during vector averaging and fusion, as well as reducing the size of word embedding vocabulary. The second one is whitespace normalization where we reduce multiple consecutive whitespace characters into a single space to simplify the tokenization process. Since urban petitions are full of street and political figures' names and the case with which people write them is variable, we lowercase every piece of text in order to have more consistent representations. Finally, all of the invisible or non-unicode characters get stripped from the documents.

Now, we are going to define the testing framework for our research. We split full document classification dataset after the application of preprocessing steps into two subsets: training (75%) and validation (25%). Validation dataset is used to report all of the results in the following chapter. Given the imbalances present in the dataset we use a stratified validation split, so that the distributions of labels in both training and validation subsets are similar. For the same reason we report weighted by the support of each class average F1-scores instead of accuracy which is susceptible to label imbalances. We compare the proposed model with the baseline model (without synonyms information) and with statistical natural language processing approaches which usually perform better in low-resource settings. All of the compared models had their hyperparameters tuned via grid search so that their choice does not influence the differences in metrics.

4.2 Synonyms Graph

To build a synonyms graph of Ukrainian language we used synonyms dictionary from the Official Website of the Ukrainian Language [26]. Since this is an online dictionary, some element of preprocessing was required to transform its words into the same form as the ones in the classification dataset. These preprocessing steps include: removal of sentence examples, context and abbreviations; lowercasing and deduplication. Afterwards, we created an empty undirected unweighted graph and populated it with words as nodes and if two words are synonyms, the corresponding nodes get connected with an edge. The synonyms graph statistics are shown in table 1 and its degree distribution can be seen in figure 7. From the analysis of graph properties one can conclude that synonymy in the Ukrainian language can be very deep and there exist many alternatives on how a single thought can be expressed, which further strengthens the assumption that adding synonym dictionary information to baseline models is useful for document classification. The expected quality of trained synonym embeddings is that the connected components in a graph are close in terms of a selected distance function while nodes that exist in separate subgraphs are further apart in the embedding space.

5. RESULTS

5.1 Baseline Document Embeddings

As previously mentioned, to build baseline document embeddings we need to train corresponding Word2Vec word embeddings first. This is done on the preprocessed document classification dataset and since this task is self-supervised, labels are not taken into account at this step in the pipeline. As an additional consistency precaution, we remove words that appear less than 20 times in our dataset. Word2Vec model details: output vector dimensionality – 100, learning rate – 0.025, context window – 5, training epochs – 1000, training algorithm – Adam [27], model – skipgram, objective function modification – negative sampling with 5 negative samples. Since the size of the dataset

Table 1: Synonyms graph statistics

Statistic	Value
Nodes	44664
Edges	391047
Average Degree	8.755
Number of connected components	545
Max clique size	44
Average Clustering	0.797

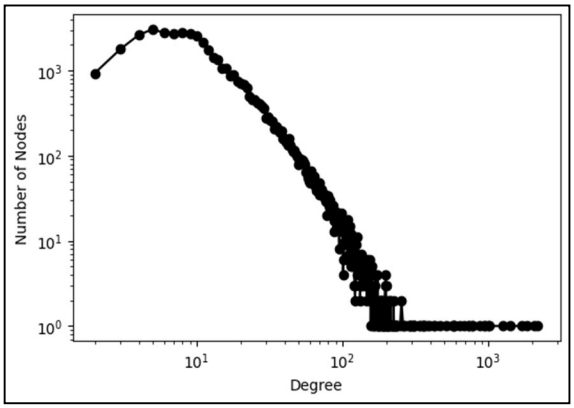

Figure 7: Synonyms graph node degree distribution

is not large, the training does not require a GPU and took around half an hour on a 2.8 GHz Intel Core i5 processor using four workers.

The resulting word vector space exhibits the properties that we were seeking, namely semantically similar words have small distances between each other, while semantically different words have large distances. Measure of distance that is frequently used in word vector spaces is cosine distance. The complement of cosine distance is cosine similarity and is defined as:

$$S(V,W) = \frac{VW}{\|V\|\|W\|} = \frac{\sum_{i=1}^{n} V_i W_i}{\sqrt{\sum_{i=1}^{n} V_i^2}\sqrt{\sum_{i=1}^{n} W_i^2}},$$

where *V* and *W* are word embeddings of size *n*. Cosine distance is then simply equal to *1 – S(V, W)*. For example, in our trained word embedding space cosine distance between words "street" and "avenue" is 0.466, while the distance between words "street" and "car" is 0.996.

After we apply our baseline model – the averaging of word vectors we get a vector space of Kyiv City petitions where the same distances logic applies. This vector space can be used for document similarity queries and clustering. We can check the Silhouette Coefficient [28] to evaluate how cohesive the clustering of petition vector space can be. For that purpose, we first cluster our baseline petition vectors with DBSCAN [29] and report the Silhouette Coefficient on the results of clustering. It takes the value of 0.468 which means that baseline documents embeddings are appropriate for clustering and capture the differences between documents relatively well.

5.2 Synonym Document Embeddings

Similarly, to build synonym document embeddings we first trained synonym node embeddings. This time, though, we do not filter out low-frequency words, because we are trying to model every node in the graph. Hyperparameters of the Node2Vec algorithm that worked well for our experiment are: output vector dimensionality – 100 (governed by the dimensionality of baseline embeddings for easier fusion), random walk length – 5, number of walks – 100, learning rate – 0.1, context window – 5, p return hyperparameter – 1, q inout hyperparameter - 1, training epochs – 5, training algorithm - Adam, model – skipgram, objective function modification – negative sampling with 50 negative samples. Training took 3 hours on the same machine where the baseline document embeddings were trained.

Since the whole concept of training node embeddings for the synonymy graph was to get dense node representations that encode synonymy, the distances between synonymic words should be significantly smaller than between random words. Our trained node embeddings show this property. A similar to the previous subsection setting but in synonym word embeddings yields the following results: distance between "street" and "avenue" is 0.121, while the distance between nodes "street" and "car" is 0.735. As a result of a qualitative analysis of the resulting node vectors we could observe that these vectors are more distinctive than Word2Vec word vectors, which makes sense because they were trained on a well-defined inherently clustered structure instead of real texts where synonymy can only be captured by tracking contexts which are very limited in a low-resource setting.

Document embeddings based on synonyms node embeddings also exhibit promising behavior with a Silhouette Coefficient equal to 0.863. This indicates that the synonyms-based document embedding space can be clustered easily and since this score is significantly different from baseline embeddings, might give an edge in the document classification task.

5.3 Document Classification

After both baseline and synonym document vectors are ready, we can start document classification. Because of its predictive power and non-linear qualities, a simple neural network – a multilayered perceptron was chosen as the classification method. We train and compare four classifiers with different document representations, namely: TF-IDF, baseline Word2Vec average of word vectors, baseline vectors concatenated with synonyms vectors, and baseline vectors fused by a weighted sum with synonyms vectors. We perform

hyperparameters search for every model-vector combination to limit the possibility of random initialization or extra parameters influencing the reported scores and the nature of positive impact. The resulting metrics are shown in Table 2. From it we can see that the weighted sum method outperforms other classification procedures by at least 2% while the concatenation method does not yield significant improvements over the baseline.

The difference between confusion matrices of the baseline and our best models is shown in figure 8. Positive values on the diagonal and negative values on non-diagonal places correspond to improvements in classification of a certain class. Considering the label distribution in the dataset, it can be stated that the enhanced model helped capture some nuance in low-frequency classes, but improved better represented labels classification only minorly.

5.4 α and β Selection

Since the values of α and β have a great influence on the performance of our method, we include the analysis of their optimal values. To deduce them we ran a grid search algorithm with full pipeline retraining and validation. We varied both α and β values from -1.5 to 1.5 in 0.02 increments, which resulted in 22500 experiments. The results are presented in figure 9. Smaller absolute values of β together with larger absolute values of α result in higher weighted F1-scores in our experimental setting which corresponds to the assumption that synonym information is only an enhancement of baseline vectors. Moreover, synonym embeddings are not sufficient to completely replace baseline embeddings as they lack the specificity and details that are present in practical applications. For example, street names cannot be captured by synonym embeddings since it is impossible for them to have synonyms. This can be seen in the area of the space where α is close to zero and β is varied. The absolute maximum of metrics that were achieved during our experiments is described by $\alpha = 1.4$ and

Table 2: Scores of different document representation approaches on document classification

Approach	Weighted F1-Score
TF-IDF	0.639
Baseline (Word2Vec averaging)	0.637
Baseline concatenated with synonym embeddings	0.640
Weighed sum of baseline and synonym embeddings	0.659

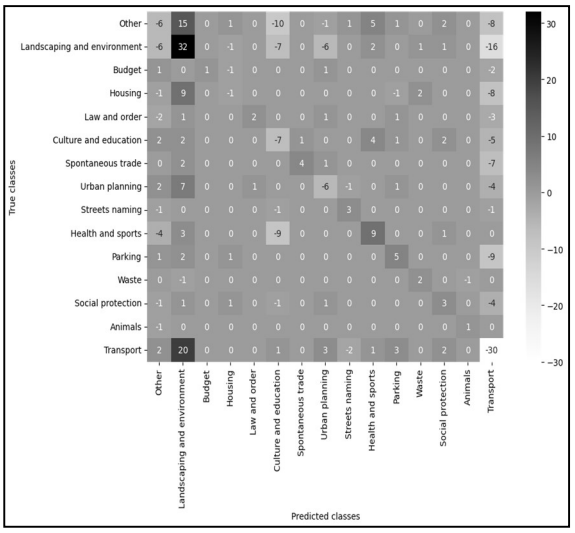

Figure 8: Baseline and weighted-sum fusion synonyms embeddings confusion matrices difference

$\beta = 0.14$. We also provide the results of the abovementioned grid search with a different metric – macro F1, which does not take label imbalance into account, in figure 10 and the space of optimal weighted sum fusion hyperparameters looks similar.

One possible explanation for the positive results of our study could simply be a regularization effect of noise that is introduced as the synonym graph embeddings, similar to [14] and [30]. However, we believe that it is not the case for two reasons: (1) we experimented with adding noise to the baseline document embedding framework and it did not improve the results; (2) if it were noise-related regularization, we would see a symmetrical decay of scores around $\alpha = 1.0$ while varying β hyperparameter.

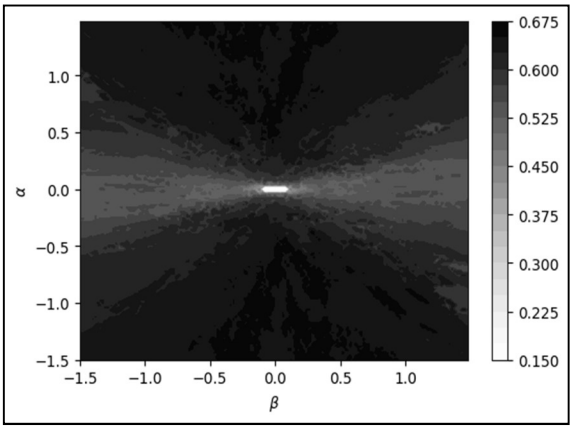

Figure 9: α and β hyperparameters weighted F1-score impact

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

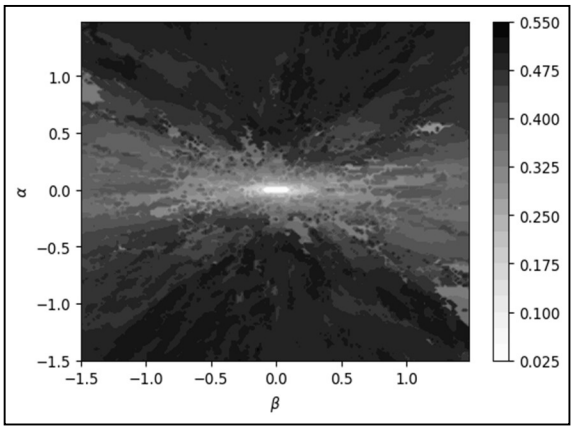

Figure 10: α and β hyperparameters macro F1-score impact

6. CONCLUSION

The research in this paper focused on the task of document representation and classification in low-resource settings. We propose a way to increase the predictive power of baseline vector-based models by incorporating additional synonymy information explicitly. This can be done via fusion of baseline document embeddings with node embeddings of a language synonyms graph. To build such node embeddings a random walk-based graph embedding method can be applied and we show Node2Vec to be successful in this regard. The main assumption of our research is that language synonym dictionaries are available, which may not be the case in all low-resource settings, for example, in languages where the linguistic science and community is still developing.

The described method falls into the category of embedding-based transfer learning low-resource natural language processing methods because it transfers the information from a different source, namely synonyms dictionary, via graph embeddings to the document classification task.

Our experiments tested two fusion techniques: concatenation and weighted summation. To ensure a low-resource setting we used a dataset of Kyiv City petitions written in Ukrainian which has a limited amount of extra high-quality datasets to reuse information from. Ukrainian synonyms dictionary provided the means to build an undirected graph from which embeddings were learned by the Node2Vec algorithm. The developed model resulted in a 2% increase of weighted F1-scores over the baseline model when weighted summation is used. Concatenation of baseline and synonyms document vectors did not show significant improvements to be useful. We also provide an analysis of weighted summation hyperparameters and list optimal values of weights for achieving the best results.

Future research can go into several directions. Firstly, it is possible to check the impact of other graph embedding approaches and algorithms on the resulting metrics and compare them. Since the size of synonym dictionaries is relatively stable, this investigation can also explore the time needed to build synonym embeddings, as random walk-based methods usually take a lot of time to train and are very sensitive to hyperparameters. Secondly, one can explore the fusion with other types of linguistic information that is incorporated in a graph-like form, e.g., etymological or referential meaning dictionaries. If this is proven to have a positive impact on the document classification performance, the next logical step would be to combine several graph-based embeddings and check if this enhances the method further. Last but not least, the approach could be tested in other low-resource settings such as underrepresented on the Web languages different from Ukrainian.

REFERENCES:

[1] Devlin J, Chang MW, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805. 2018 Oct 11.

[2] Zhu Y, Kiros R, Zemel R, Salakhutdinov R, Urtasun R, Torralba A, Fidler S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. InProceedings of the IEEE international conference on computer vision 2015 (pp. 19-27).

[3] Harris ZS. Distributional structure. Word. 1954 Aug 1;10(2-3):146-62.

[4] Luhn HP. A statistical approach to mechanized encoding and searching of literary information. IBM Journal of research and development. 1957 Oct;1(4):309-17.

[5] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781. 2013 Jan 16.

[6] Bengio Y, Ducharme R, Vincent P, Janvin C. A neural probabilistic language model. The journal of machine learning research. 2003 Mar 1;3:1137-55.

[7] Xing C, Wang D, Zhang X, Liu C. Document classification with distributions of word vectors. InSignal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific 2014 Dec 9 (pp. 1-5). IEEE.

[8] Hedderich MA, Lange L, Adel H, Strötgen J, Klakow D. A survey on recent approaches for natural language processing in low-resource scenarios. arXiv preprint arXiv:2010.12309. 2020 Oct 23.

[9] Perez L, Wang J. The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621. 2017 Dec 13.

[10] Mintz M, Bills S, Snow R, Jurafsky D. Distant supervision for relation extraction without labeled data. InProceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP 2009 Aug (pp. 1003-1011).

[11] Karamanolakis G, Mukherjee S, Zheng G, Hassan A. Self-Training with Weak Supervision. InProceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies 2021 Jun (pp. 845-863).

[12] Kumar V, Choudhary A, Cho E. Data Augmentation using Pre-trained Transformer Models. InProceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems 2020 Dec (pp. 18-26).

[13] Eykens J, Guns R, Engels T. Clustering social sciences and humanities publications: Can word and document embeddings improve cluster quality?. InProceedings of the 18th conference of the International Society for Scientometrics and Informetrics 2021 (Vol. 1, pp. 369-374).

[14] Wei J, Zou K. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. arXiv preprint arXiv:1901.11196. 2019 Jan 31.

[15] Yamada I, Asai A, Sakuma J, Shindo H, Takeda H, Takefuji Y, Matsumoto Y. Wikipedia2Vec: An Efficient Toolkit for Learning and Visualizing the Embeddings of Words and Entities from Wikipedia. InProceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations 2020 Oct (pp. 23-30).

[16] Harris R. Synonymy and linguistic analysis. University of London, School of Oriental and African Studies (United Kingdom); 1970.

[17] Yan S, Xu D, Zhang B, Zhang HJ. Graph embedding: A general framework for dimensionality reduction. In2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) 2005 Jun 20 (Vol. 2, pp. 830-837). IEEE.

[18] Goyal P, Ferrara E. Graph embedding techniques, applications, and performance: A survey. Knowledge-Based Systems. 2018 Jul 1;151:78-94.

[19] Grover A, Leskovec J. node2vec: Scalable feature learning for networks. InProceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining 2016 Aug 13 (pp. 855-864).

[20] Grohe M. word2vec, node2vec, graph2vec, x2vec: Towards a theory of vector embeddings of structured data. InProceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems 2020 Jun 14 (pp. 1-16).

[21] Zhang C, Yang Z, He X, Deng L. Multimodal intelligence: Representation learning, information fusion, and applications. IEEE Journal of Selected Topics in Signal Processing. 2020 Apr 15;14(3):478-93.

[22] Kyiv City electronic petitions. Retrieved from: https://petition.kyivcity.gov.ua/ (September 06, 2020).

[23] Usage statistics of content languages for websites. Retrieved from: https://w3techs.com/technologies/overview/content_language (October 5, 2021).

[24] Joulin A, Grave E, Bojanowski P, Mikolov T. Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759. 2016 Jul 6.

[25] Joulin A, Grave E, Bojanowski P, Douze M, Jégou H, Mikolov T. Fasttext. zip: Compressing text classification models. arXiv preprint arXiv:1612.03651. 2016 Dec 12.

[26] Dictionary of synonyms on the Official website of the Ukrainian Language. Retrieved from: https://ukrainskamova.com/ (June 15, 2021).

[27] Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980. 2014 Dec 22.

[28] De Amorim RC, Hennig C. Recovering the number of clusters in data sets with noise features using feature rescaling factors. Information sciences. 2015 Dec 10;324:126-45.

[29] Ester M, Kriegel HP, Sander J, Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. Inkdd 1996 Aug 2 (Vol. 96, No. 34, pp. 226-231).

[30] Zhu C, Cheng Y, Gan Z, Sun S, Goldstein T, Liu J. FreeLB: Enhanced Adversarial Training for Natural Language Understanding. InInternational Conference on Learning Representations 2019 Sep 25.