Budapesti Mûszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Department of Automation and Applied Informatics

Universal embeddings

Diplomaterv

Készítette
Eszter Iklódi

Konzulens
Gábor Recski

May 3, 2018

Contents

Kivonat **4**

Abstract **5**

1 Introduction **6**
1.1 Natural Language Processing 6
1.1.1 Common tasks of NLP 7
1.1.2 Motivation for NLP research 7
1.2 Thesis objectives 8
1.3 Thesis results 8
1.4 References 8
1.5 Document structure 9

2 Word embeddings **10**
2.1 Semantic encoding of words 10
2.2 State-of-the-art models for learning word embeddings 11
2.3 Multilingual word embeddings 12
2.3.1 Motivation 12
2.3.2 Tasks 13
2.3.3 Applications 16
2.4 State-of-the-art multilingual embedding models 16
2.4.1 First attempt: Mikolov et al. 17
2.4.2 Various improvements 17
2.4.3 Without parallel data 19

3 Proposed model **21**
3.1 Multilingual data 21
3.1.1 English-Italian setup of Dinu 21
3.1.2 The `fastText` embedding 22
3.1.3 Panlex 24
3.2 Description of our method 26
3.2.1 Cosine similarity and precision 27
3.2.2 Equation to optimize 27
3.2.3 Configuration parameters 28

3.3 Software architecture . . . 30
3.3.1 Implementation . . . 30
3.3.2 Brief description of the most important used packages . . . 32

4 Experiments **33**
4.1 Baseline experimental setting . . . 33
4.1.1 Adjusting basic parameters . . . 34
4.1.2 Experimenting with SVD . . . 36
4.2 Dinu's experimental setting and our baseline system . . . 40
4.2.1 `fasttext` . . . 40
4.2.2 Dinu's word vectors . . . 41
4.3 Panlex experiments . . . 42
4.3.1 Data inspection . . . 42
4.3.2 Training on PanLex . . . 43
4.3.3 Training on Dinu, testing on PanLex . . . 45
4.4 Continuing the baseline system with PanLex data . . . 46

5 Conclusions and future work **48**
5.1 Summarizing the contributions of the thesis . . . 48
5.2 Future word . . . 48

Köszönetnyilvánítás **49**

List of Figures **50**

List of Tables **51**

Acronyms **52**

Bibliography **57**

HALLGATÓI NYILATKOZAT

Alulírott *Eszter Iklódi*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot/ diplomatervet (nem kívánt törlendõ) meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelmûen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerzõ(k), cím, angol és magyar nyelvû tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhetõ elektronikus formában, a munka teljes szövegét pedig az egyetem belsõ hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetõvé.

Budapest, May 3, 2018

Eszter Iklódi
hallgató

Kivonat

Mindennapi életünkben egyre fontosabb szerepet tölt be a természetes nyelv számítógép segítségével történő feldolgozása. Digtitalizált világunkban egyre inkább alapkövetelmény, hogy a gép és ember közötti kommunikáció természetes nyelven történjen. Ennek a megvalósításához elengedhetetlen az emberi nyelv szemantikai értelmezése.

Manapság a state-of-the-art rendszerekben a szavak szemantikai reprezentációja sokdimenziós vektorokkal, word embedingek-kel történik. Diplomaterv munkámban már feltanított word embeddingek-hez keresek olyan fordítási mátrixokat, amelyek képesek egy adott nyelvű word embedding univerzális térbe történő leképzésére.

A rendszert először Dinu angol-olasz benchmark adatán [28] tanítjuk, majd pedig a PanLex adatbázisból [11] kinyert angol-olasz fordítási párokon kísérletezünk. Végül a két adat kombinálásával is futtatunk kísérleteket.

Dinu adatán futtatott kísérleteink eredményei habár elmaradnak a jelenlegi state-of-the-art rendszerek teljesítményétől, azonban messze meghaladják Mikolov baseline rendszerének eredményeit [42], továbbá összemérhető teljesítményt nyújtanak Faruqui [29] és Dinu [28] szofisztikáltabb rendszereinek teljesítményével.

A PanLex adatbázison futatott kísérleteink eredményei több, mint egy nagyságrenddel alulmúlják a Dinu adaton futtatott kísérleteink eredményeit. Ezen az adaton különböző kísérleti beállítások ellenére sem sikerült jelentős javulást elérni. Mindazonáltal a Dinu adaton tanított rendszerünk PanLex adattal történő továbbtanításakor az olasz-angol irány precision számai enye javulást mutattak.

Abstract

Computer-driven natural language processing plays an increasingly important role in our everyday life. In our digital world using natural language for human-machine communication has become a basic requirement. In order to meet this requirement it is inevitable to analyze human languages semantically.

Nowadays, state-of-the-art systems represent word meaning with high dimensional vectors, i.e. with word embeddings. In my thesis work I am searching for translation matrices to pre-trained word embeddings, such that the translation matrices will be able to map these embeddings into a universal space.

First we train our system on Dinu's English-Italian benchmark data [28], then we experiment on English-Italian word pairs extracted from the PanLex database [11]. Finally, we run some other experiments combining these two data sources.

Although our results obtained by using Dinu's data are worse than state-of-the-art results on this data, they perform significantly better than Mikolov's baseline system [42], and they provide a comparable performance with Faruqui's [29] and Dinu's [28] more sophisticated systems.

Results of the experiments run on the PanLex database are more than one order lower, than our results obtained by using Dinu's data. Despite the numerous attempts with different configuration settings, we did not manage to reach a significant improvement on this data. Nonetheless, when continuing the training process of our system trained on Dinu's data with PanLex entries, we observed a slight improvement on the Italian-English precision numbers.

Chapter 1

Introduction

1.1 Natural Language Processing

In this section I summarize the main motivations and tasks of the field of Natural Language Processing (NLP).

NLP is a vibrant interdisciplinary field with many different names, all reflecting a different facet of it. It is often referred to as speech and language processing, human language technology, computational linguistics, or speech recognition and synthesis. The main goal of this field is to get computers to be able to use human languages as a communication protocol between machines and human users.

NLP is difficult since it deals with what is considered to be one of the most delicate characteristics of human beings: with human languages. This field is strongly connected with artificial intelligence since the way humans conceive the world is mainly happening in terms of human languages.

Being nowhere near as fast as digital channels, expressing ourselves by means of human languages is a very effective way of communication, though. Saying only the minimum message our listeners can fill up the rest with their world and common knowledge, and can easily figure out the missing or misunderstood parts from the context of the situation. This way they are also able to resolve ambiguities, homonyms etc. without even noticing it. Nonetheless, these tasks for a computer are not that trivial at all.

The importance of computer integrated human language communication has gone as far as assigning truly intelligent machines to the ability of being capable of processing language as skillfully as humans do. This idea was first introduced by Alan Turing (1950) who proposed what has come to be known as the Turing test.

To get a more detailed overview of what NLP is about, for interested readers I recommend Dan Jurafsky's *Speech and language processing* book [36]. For those who prefer video lectures I advise checking out the *Natural Language Processing with Deep Learning* course held by Christopher Manning and Richard Socher, professors of the Stanford University School of Engineering. This course is available for free on YouTube [1].

1.1.1 Common tasks of NLP

NLP comprises a wide variety of tasks. Some of them like spam detection, part-of-speech (POS) tagging, or named entity recognition are considered to be mostly solved problems. Applications for these tasks are now out in the market and are usually integrated to our smart devices even by default.

With some other tasks great progress has been made recently which implies the existence of already fair enough applications but means that research work is still has to be done. Among them there are tasks like sentiment analysis, words sense disambiguation, syntactic parsing, machine translation etc.

What is still considered to be very hard is to understand the meaning of a text. There are numerous interesting tasks for example question answering, dialogues, summarization, paraphrases, or text inference, for which in order to make relevant progress dealing with the semantics is inevitable.

1.1.2 Motivation for NLP research

Nowadays NLP technologies are becoming more and more integrated into our every life. With the advent of smart phones the importance of language has gone even further. These devices are having small and rather inconvenient keyboards, thus speech-driven communication seems very appealing. Big companies like Amazon, Apple, Facebook, Google, etc. are all releasing products that use natural languages (human languages) to communicate with users. Since the contributions of this thesis are aiming the research field of word meaning and universal semantic representations, below I will only list applications that can directly take advantage of these contributions.

Speech-driven assistance applications can make our everyday life more enjoyable, more comfortable and more convenient. They already help children with developing delicate skills and they provide an immense help for elderly people or people living with disabilities. These systems include speech input for which at first automatic speech recognition technologies should be applied. But after that, in order to understand the goal of the user, we must run a semantic analysis as well.

An early version of conversational agents and certain strongly domain-based chatbots are already out on the market, providing 24 hour, immediate assistance for customers. By letting computers do the monotone and not at all creative tasks employees could have more interesting jobs, jobs that only humans can do, or they could have less working hours in a week. Either of them would be a great progress for the society [2].

Advances in machine translation has already created a world where non-English speakers can also enjoy the benefits of the English-based web services. Generally, we can say that for widespread languages machine translation has already reached a fairly usable state, for rare languages, however, it is still facing difficulties.

There are also numerous Web related tasks that are strongly relying on the semantic analysis of the text. One promising task is for example the Web-based question answering task which is basically an extended version of the classical Web search, whereby instead

of searching just for key words it would also be possible to ask complete questions and thus communicate with the search engine just like human beings do [48]. For all these applications, however, it is inevitable to look way behind the syntactic surface and dig deep into the underlying semantics.

1.2 Thesis objectives

In our work we are examining word meaning. Given the need for robust representations for many languages, the question of whether human conceptual structure is universal has recently gained interest not only among cognitive scientists ([51], [39], [33]), but among computational linguists as well. Youn et al. [60] has shown that human conceptual structure is independent of certain non-linguistic factors such as geography, climate, topology or literary traditions. Based on such findings we propose a procedure to construct a universal semantic representation in form of translation matrices that serve to map each language to a universal space. We use the pre-trained fastText word embedding [26] (discussed in 3.1.2), which contains word vectors for 294 languages. During the training process we align a set of word translation pairs extracted from various gold dictionaries (Dinu's data: discussed in 3.1.1 and the PanLex database: discussed in 3.1.3).

1.3 Thesis results

We train and test our system using the `fasttext` pre-trained embedding and various gold word translation sets. The experiments and results are discussed in more detail in Chapter 4.

First, we use both for training and testing Dinu's benchmark English-Italian word translation data [28]. With our proposed method we obtain significantly better results, both in English-Italian and in Italian-English directions, than Mikolov's baseline system [42]. Furthermore, we obtain comparable results with Faruqui's [29] and Dinu's [28] more elaborated systems on the same benchmark data. This system we call our baseline system. For more details see 4.2.

Next, we train our system on English-Italian word translation pairs extracted from the PanLex database [11]. Our baseline system gives more than one order lower results on PanLex, than on Dinu's data. Even after trying out various configuration settings, the obtained results still do not get significantly higher. For more details see 4.3.

Finally, we use the extracted PanLex word translation pairs for continuing the training of our baseline system. One surprising finding is that this model reaches a slightly better performance on Italian-English direction, than our baseline system. For more details see 4.4.

1.4 References

The code of our system is available on Github on the following link:
https://github.com/Eszti/dipterv

The whole code base was implemented by myself except for an earlier version of the script which extracts translation pairs from the PanLex database, found here: https://github.com/Eszti/dipterv/blob/master/panlex/scripts/panlex/extract_tsv.py. This piece of code was implemented by my supervisor, Gábor Recski.

1.5 Document structure

The thesis is structured as follows:

- **Chapter 1** briefly explains the goals and motivations of the research field of NLP. It also summarizes the main contributions and the results of the accomplished work.
- **Chapter 2** discusses the state-of-the-art semantic word representations, the word embeddings. It briefly presents the standard learning procedure for monolingual word vectors (word2vec) and it introduces the concept of multilingual word-embeddings.
- **Chapter 3** describes the available resources for multilingual embedding learning that were utilized during this work. It also introduces the proposed model in detail. It explains the learning procedure and the basic infrastructural and architectural features of our system.
- **Chapter 4** presents all the experiments. It summarizes our results and compares them with the performance of other systems.
- **Chapter 5** is devoted to the description of the future work. This chapter suggests modifications, follow-ups for which we either didn't have time to accomplish, or which are beyond the scope of this thesis work.

Chapter 2

Word embeddings

2.1 Semantic encoding of words

Within the field of natural language processing a more specific area concentrates on semantic representations which are being leveraged both by classical semantic tasks such as question answering or chatbots and by other NLP (Natural Language Processing) tasks which in the strict sense of the word are not considered semantic tasks such as machine translation or syntactic parsing. A crucial part of all semantic tasks is to have a proper word representation which is capable of encoding the meaning as well.

One way to build a semantic representation is to use a distributional model. The idea is based upon the observation that synonyms or words with similar meanings tend to occur in similar contexts, or as it was articulated by Firth in 1957: *You shall know a word by the company it keeps* [32]. For example in the following two sentences ‘‘`The cat is walking in the bedroom`’’ and ‘‘`A dog was running in a room`’’ words like ‘‘`dog`’’ and ‘‘`cat`’’ have exactly the same semantic and grammatical roles therefore we could easily imagine the two sentences in the following variations: ‘‘`The dog is walking in the bedroom`’’ and ‘‘`A cat was running in a room`’’ [25]. Based upon this intuition, what distributional models are aiming to do is to compute the meaning of a word from the distribution of words around it [36]. The obtained meaning representations are usually high dimensional vectors, called as word embeddings, which refer to their characteristic feature that they model a world by embedding it into a vector space.

Embeddings are not only proved to be a better alternative of n-grams used for language modelling [25], but they are doing quite well on semantic tasks too. Mikolov et al. [43] have shown that the characteristics of word embeddings go way beyond the simple syntactic regularities. They showed that applying simple vector operations (e.g. vector addition and subtraction) can often produce meaningful results. For example it was shown that *vector(”King”) - vector(”Man”) + vector(”Woman”)* results in a vector that is closest to the vector representation of the word *Queen* [44]. Moreover, state-of-the-art results on work similarity tasks are all held by word embeddings, where the similarity of two words are calculated as the normalized dot product of the corresponding word vectors. This measure is the so-called cosine similarity of word embeddings.

Another way to build semantic representations is to utilize lexical databases. In some previous works of the research team we created a hybrid system leveraging both the `4lang` othologycal model described in [37], [38] and [19] and various distributional models, i.e. various word embeddings. With this system in 2016 we reached a state-of-the-art score on the `SimLex-999` [35] benchmark data [50].

In the following sections I will describe the basic procedure of training word embeddings and following that I will focus on a more specific field of computational semantics, on multilingual word embeddings.

2.2 State-of-the-art models for learning word embeddings

In 2003 Bengio et al.[25] suggested a probabilistic feedforward neural network language model (NNLM) for learning a distributed representation for words. The network consisted of input, projection, hidden and output layers, where at the input layer the N previous words are encoded using 1-of-V coding, where V is the size of the vocabulary. Figure 2.1 shows an overview of the architecture. This procedure was evaluated in terms of perplexity at which in comparison with the best of the n-grams it proved to be much better. The drawback of this architecture is that it becomes complex for computation between the projection and the hidden layer, as values in the projection layer are dense.

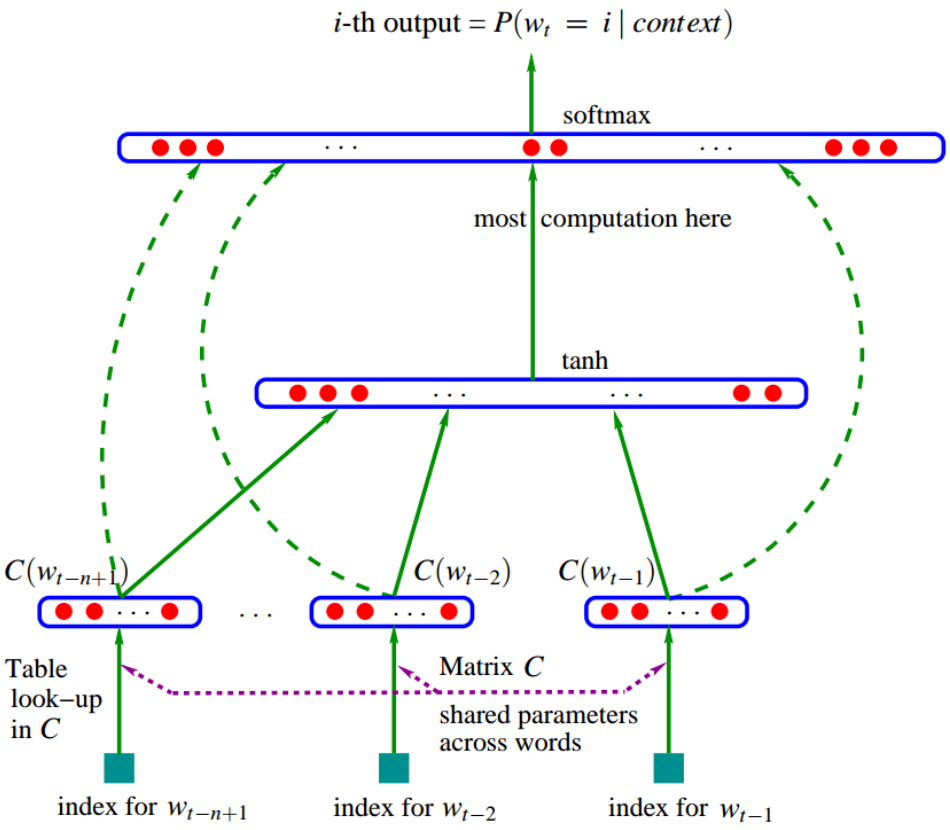

Figure 2.1: *Network architecture proposed by Bengio et al.[25]*

10 years later, in 2013 Mikolov suggested a Bag-oF-Words Neural Network, more specif-

ically two different architectures [41]. The first one, denoted as the CBOW (Continuous Bag-of-Words Model) tried to predict the current word based on the context, whereas the second one, denoted as the continuous skip-gram model tried to maximize the classification of a word based on another word in the same sentence. Both models worked better than the NNLM suggested by Bengio [25] both on semantic and syntactic tasks, while between the two models of Mikolov the CBOW turned out to be slightly better on syntactic tasks and the skip-gram on semantic tasks. Mikolov's procedure has become known as the `word2vec` procedure and the source code is available on github http://deeplearning4j.org/word2vec. The architecture of the CBOW and the skip-gram models are shown in figure 2.2.

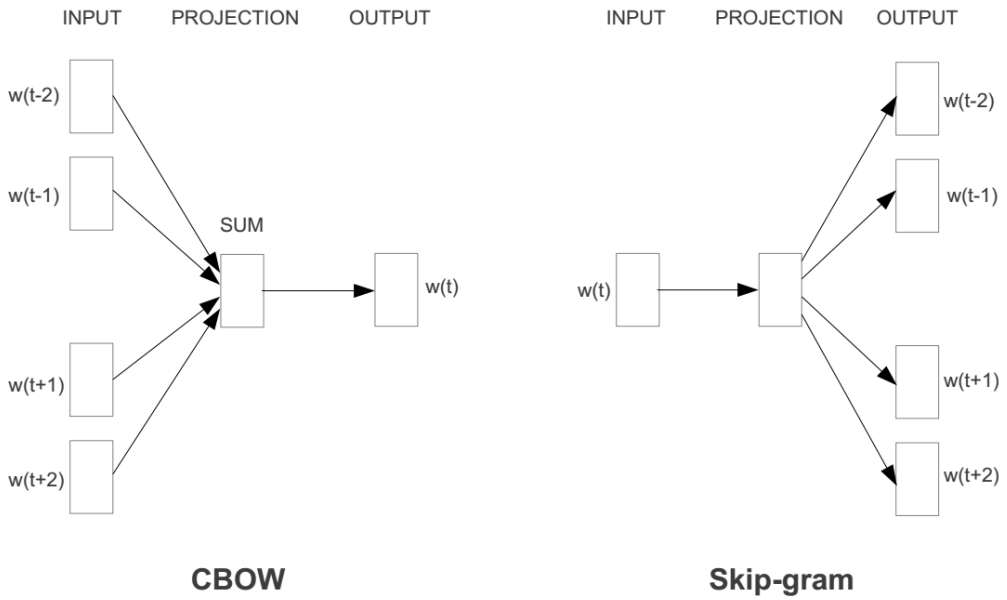

Figure 2.2: *Bag-of-words neural networks suggested by Mikolov et al.[43]*

Embeddings are usually evaluated on word similarity and word analogy tasks. Besides providing quite promising results on them, they have also been applied to many downstream tasks, from named entity recognition and chunking [58] to dependency parsing [23]. It has furthermore been shown that weakly supervised embedding algorithms can also lead to huge improvements for tasks like sentiment analysis [56].

2.3 Multilingual word embeddings

In this section I describe the importance of multilingual word embeddings. I also explain how it is possible to incorporate word embeddings trained on monolingual text corpora into a multilingual context. After that I present a brief summary about the previous attempts on constructing cross-lingual word vector representations.

2.3.1 Motivation

The question how to model representations is a highly interdisciplinary issue to discuss. Within cognitive science, traditionally there are two dominating approaches to this prob-

lem. The first one is the *symbolic* which states that cognitive systems can be described as Turing machines. The second one, denoted as *associationism*, says that representations are associations among different kinds of information elements. In his book, *Conceptual Spaces: The Geometry of Thought* [33], Gärdenfors advocates a third approach, which he calls *conceptual* from. This representation is based on using geometrical structures rather than symbols or connections among neurons.

To go a step further one could ask whether these structures are universal among all human beings. Regarding this question with the eyes of a computer scientist we might form this problem as whether it is possible to model meaning universally, i.e. language independently. Current meaning representations are leaned from monolingual corpora, therefore they infer language dependency. But is there a way to find one single representation instead of a different one for each and every human language?

Youn et al. [60] suggested that human brain may reflect distinct features of cultural, historical, and environmental background in addition to properties universal to human cognition. They provided an empirical measure of semantic proximity between concepts using entries of the Swadesh list [55]. The Swadesh list is cross-linguistic dictionary which includes a 110 and a 207 long list of basic concepts in approximately 2000 languages. Youn et al. took 22 concepts of this list that refer to material entities (e.g. STONE, EARTH, SAND, ASHES), celestial objects (e.g., SUN, MOON, STAR), natural settings (e.g., DAY, NIGHT), and geographic features (e.g., LAKE. MOUNTAIN). Then, they applied translation and back-translation through various languages. As a result of numbers of polysemies in the resulting graph originally distinct concepts become connected. For example the Spanish word *cielo* in English both means *heaven* and *sky*. Thus by applying English-Spanish-English translation and back-translation the two English words *heaven* and *sky* become connected. The more such polysemous words we find, the stronger this connection becomes. For example if besides Spanish, we also apply the translation and back-translation through German, the same polysemy appears: the German word *Himmel* in English both means *heaven* and *sky*, just like the Spanish *cielo* word. The procedure is shown on figures 2.3 and 2.4.

Statistical analysis of the obtained graphs constructed over the polysemies observed in the above-mentioned 22-word-long subset of basic vocabulary showed that the structural properties of these graphs are consistent across different language groups, and largely independent of geography, environment, and the presence or absence of literary traditions. Based upon these findings we assume that meaning, at least to a certain extent, is universal, thus representing semantics at universal level is reasonable.

2.3.2 Tasks

Beyond the theoretical level of whether meaning is universal there are numerous practical problems for which cross-lingual embeddings might come in handy. In this section I write about the different tasks where solutions can be facilitated by utilizing multi-lingual embeddings.

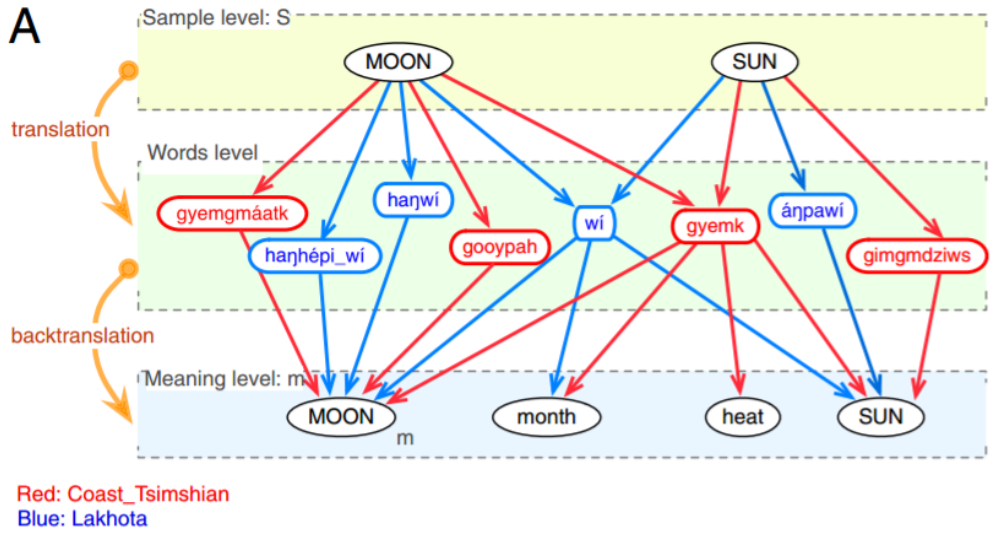

Figure 2.3: *Translating MOON and SUN through polysemous words.*

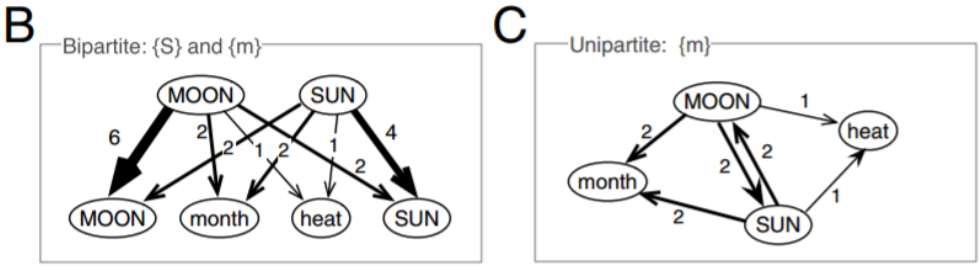

Figure 2.4: *Making links between English concepts through eliminating the internal nodes.*

Cross-language part-of-speech tagging

POS-tagging (part-of-speech tagging) is the task for annotating a text with part-of-speech tags. The fundamental idea behind the multilingual learning of part-of-speech tagging is that when assigning part-of-speech tags the patterns of ambiguity differ across languages. A word with part-of-speech tag ambiguity in one language may correspond to an unambiguous word in the other language. For example, the word “can” in English may function as an auxiliary verb, a noun, or a regular verb, however, translating the sentence into other languages the different meanings of "can" are likely to be expressed with different lexemes. By combining natural cues from multiple languages, the structure of each POS-tagger becomes more apparent [46].

Cross-language super sense tagging

The Semantic Web paradigm is often required to provide a structured view of the unstructured information expressed in texts. In order to cover the web scale in almost any languages, abundance of such knowledge is required. More specific fields of NLP such as ontology learning and information extraction are focusing on finding solutions to this difficult problem.

SuperSense Tagging is the problem of assigning "supersense" categories (e.g. `person`,

act) to the senses of words according to their context in large scale texts. Opposite to NER (Named Entity Recognition) systems a Super Sense Tagger does not make a difference between proper and common names. These "supersense" categories include 41 general concepts defined by WordNet [24], which originally introduces 45 lexicographer's categories [30]. The idea behind WordNet is to provide lexicographers an initial broad classification for lexicon entries.

Attempts for creating such systems have already been made. For example Picca et al. [47] trained a multilingual super sense tagger on the Italian and English languages. Despite the fact that they did not use any word embeddings, the introduction of multilingual word embeddings to this task could significantly facilitate the development of multilingual knowledge induction, ontology engineering, and knowledge retrieval.

Machine translation

Machine translation is the task of translating a text automatically with a computer from a source language to a target language. Current research works are focusing on finding the appropriate level of representations when performing the translations. Basic approaches are: `tree-to-string`, `string-to-string`, and `string-to-tree`, as shown in figure 2.5.

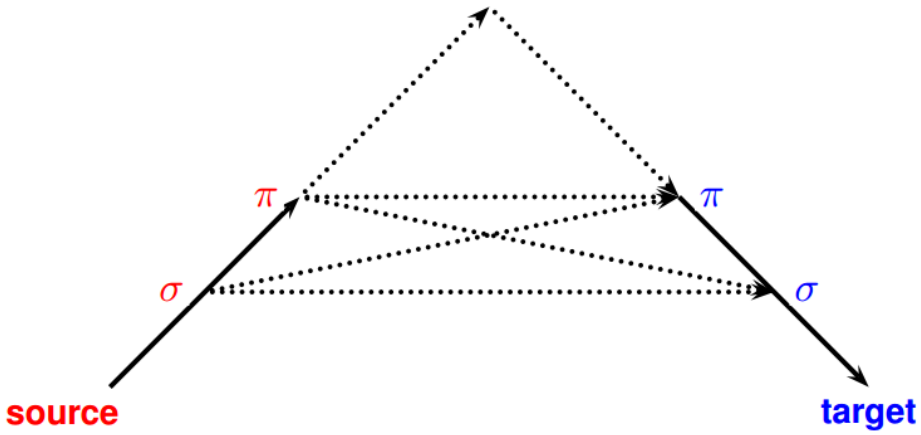

Figure 2.5: *Levels of representation in Machine Translation.* $\pi \rightarrow \sigma : tree - to - string; \sigma \rightarrow \sigma : string - to - string; \sigma \rightarrow \pi : string - to - tree.$

Translation models, however, often fail to generate good translations for infrequent words or phrases. Previous works attacked this problem by inducing new translation rules from monolingual data with a semi-supervised algorithm. Nevertheless, this approach does not scale very well since it is quite expensive computationally. Zhao et al. [61] proposed a much faster and simpler method that creates translation rules for infrequent phrases based on phrases with similar continuous representations, i.e. with similar word vectors, for which a translation is known. Their method improved a phrase-based baseline by up to 1.6 BLEU on Arabic-English translation, and it was three-orders of magnitudes faster than existing semi-supervised methods and 0.5 BLEU more accurate.

By introducing a universal vector space, in order to cover all possible translation pairs for n languages, instead of having to train $\binom{n}{2}$ translators it would be enough to train only

2n translators, for each language from the source space to the universal space and vica versa, which would significantly simplify the Machine Translation task.

Under-resourced languages

Dictionaries and phrase tables are the basis of modern statistical machine translation systems. Mikolov et al. [42] showed a method that can automate the process of generating and extending dictionaries and phrase tables. They could translate missing word and phrase entries by learning language structures based on large monolingual data and mapping between languages from small bilingual data. This is a powerful opportunity for rare languages to join the mostly English-based world of the Web and for non-English speakers to enjoy its benefits without having to speak English.

2.3.3 Applications

Facebook has already made use of multilingual embeddings [16]. To better serve their community they offer features like Recommendations [5] and M Suggestions [4] in more languages. These services are based on text classification, which refers to the process of assigning a predefined category from a set to a document of text. With language-specific NLP techniques supporting a new language implies solving the problem once again, from scratch. One way is to train a separate classifier for each language, which means collecting every time a separate, large set of training data. Collecting data is an expensive and time-consuming process, which becomes increasingly difficult when scaling it up to support more than 100 languages. Another way is to train only one classifier (e.g. an English one) and then, before applying this classifier for languages different from English, as a preprocessing step, the text first will be translated in English. This solution is prone to error propagation and, in addition, it involves an additional call to the translation service which leads to a significant degradation in performance.

Using multilingual embeddings to help to scale to more languages is a great advantage. Since the words in the new language will appear close to the words in trained languages in the embedding space, the classifier will be able to do well on the new languages as well. It is not necessary to call translation services, so it does not affect the performance either.

2.4 State-of-the-art multilingual embedding models

In this section I present a brief history on cross-lingual word vector representations. As a baseline approach I describe the procedure of Mikolov et al. [42] from 2013 and next I study various attempts made since then to improve this baseline system and to alleviate its errors. Finally, I summarize some recent attempts for obtaining multilingual word embeddings without using any parallel data.

2.4.1 First attempt: Mikolov et al.

Right after publishing their `word2vec` procedure, Mikolov et al. [42] went even further by noticing that continuous word embedding spaces exhibit similar structures across languages. They applied a simple two-step procedure:

- firstly, monolingual models of languages using huge corpora were built, e.g. by using the `word2vec` method
- secondly, a small bilingual dictionary was used to learn linear projection between the languages. These words are often referred to as anchor points.
- finally, at test time, the translation of any word from the source language is possible by projecting its vector representation from the source language space to the target language space. Once the vector in the target language space is obtained, the most similar word vector can serve as the output of the translation.

With applying only the translation matrices they achieved **51% precision@5** for translation of words between **English and Spanish**. To obtain dictionaries first they created monolingual corpora from the **WMT11** text data [17]. Then they took the most frequent words from these monolingual source datasets, and translated them using on-line **Google Translate (GT)**. Beside simple words, they also used short phrases as the dictionary entries.

In addition to the promising result on the English-Spanish word translation task, this method seemed to be working even for distant language pairs like English and Vietnamese.

In this work for the first step I used a pretrained word embedding described in 3.1.2. My work concentrates on finding the linear projections and on the evaluation of these projections.

2.4.2 Various improvements

Faruqui and Dyer

Since Mikolov's experiments various attempts were made to improve the cross-lingual embeddings. Faruqui and Dyer [29] intended to gain information from the translation of a given word in other languages. The most obvious solution would be to append the two word vectors coming from the two languages, but this procedure is highly exposed to drawbacks such as increases in dimension, introduction of irrelevant data, and incapacity of generalization across languages. To counter these problems they used **canonical correlation analysis (CCA)** which is a way of measuring the linear relationship between two multidimensional variables. It finds two projection vectors, one for each variable, that are optimal with respect to correlations, with preserving or even reducing the dimansionality. This way they obtained multi-lingual embeddings based on monolingual embeddings, which they tested on four different standard **word similarity tasks**:

- On the **WS-353** dataset [31] that contains 353 pairs of English words that have been assigned similarity ratings by humans. This dataset was later further divided into two

different fragments *similarity*, **WS-SIM**, and *relatedness*, **WS-REL** by Agierre et al. [20] who claimed that these two are different kinds of relations and should be dealt with separately

- On the **RG-65** dataset which contains 65 pairs of nouns ranked by humans [52].
- On the **MC-30** dataset which contains 30 pairs of nouns ranked by humans [45].
- On the **MTurk-287** dataset [49] that constitutes of 287 pairs of words that has been constructed by crowdsourcing the human similarity ratings using Amazon Mechanical Turk.

These word representations obtained after using multilingual evidence performed significantly better on the above mentioned evaluation tasks compared to the monolingual vectors. The method was more suitable for semantic encoding than for syntactic encoding. As a conclusion, they showed that multilingual evidence is an important resource even for purely monolingual applications.

Xing et al.

Xing et al. [59] has shown that bilingual translation can be largely improved by **normalizing** the embeddings and by restricting the transformation matrices into **orthogonal** ones.

For a comparison, they largely followed Mikolov's settings [42] for creating an English-Spanish dictionary. After extracting the monolingual datasets from the **WMT11** corpus they selected the 6000 most frequent words in English and employed the online Google's translation service to translate them in Spanish. The resulting **6000 English-Spanish** word pairs are used to train and test the bilingual transform in the way of **cross validation**. First, they reproduced Mikolov's results and then they showed that their method outperforms those results with approximately 10 % on this English-Spanish setting.

Dinu et al.

Dinu et al. [28] studied the phenomenon of **hubs**. He showed that the neighbourhoods of the mapped vectors are strongly polluted by hubs. These vectors tend to be near a high proportion of items, and thus their correct labels will be pushed down in the neighbour list when looking up for word translations. They proposed a simple method to alleviate this problem with which they achieved consistent improvements. Furthermore, they observed that many translations were in fact correct but were not present in the gold dictionary.

The experiments were carried out on an **English-Italian** dataset created by themselves and discussed in detail in 3.1.1.

Lazaridou et al.

Lazaridou et al. [40] studied some theoretical and empirical properties of general cross-space mapping function, and tested them on cross-linguistic (word translation) and cross-modal (image labelling) tasks. By introducing **negative samples** during the learning process

they could reach state-of-the-art results on **Dinu's English-Italian** word translation task. Settings for the negatives examples were studied both by choosing them random and by choosing "intruders" which are near the mapped vector, but far from the actual gold target space vector. The "intruder" approach achieved better results, and furthermore, it gave better results after just few training epochs.

Ammar et al.
Ammar et al. [21] proposed methods for estimating and evaluating embeddings of words in **more than fifty languages** in a **single shared embedding space**. The so-called multiCluster and multiCCA methods were tested on 59 languages, while the multiSkip and translation-invariance methods only on 12 languages for which high-quality parallel data was available. For the 12 languages the bilingual dictionaries were extracted from the **Europarl** parallel corpora, while for the remaining 47 languages, dictionaries were formed by translating the **20k most common words in the English monolingual corpus with Google Translate.**

Artetxe et al.
Artetxe et al. [22] built a **generic framework** that generalizes previous works made on cross-linguistic embeddings. For evaluating the methods they used the same **English-Italian** dataset by Dinu, discussed in 3.1.1. As a conclusion they published that from the proposed methods the ones with **orthogonality** constraint and a global **preprocessing** with length normalization and dimension-wise mean centering achieved the best overall results.

Smith et al.
Smith et al. [54] also proves that translation matrices should be **orthogonal**. They apply **singular value decomposition (SVD)** to achieve this. Besides, they introduce a novel **"inverted softmax"** method for identifying translation pairs, with which they improved the precision of Mikolov. Orthogonal transformations also turned out to be more robust to noise which makes it possible to learn the transformation without expert bilingual resource by constructing a "pseudo-dictionary" from the identical character strings. For evalutaion they also used Dinu's **English-Italian** setting. In order to compare their method with the previous ones they reproduced the previous experiments both in English-Italian and Italian-English directions, and published a summary in form of a table that I present here as well, as Table 3.7 and Table 3.8. Their results achieved **state-of-the-art** scores on Dinu's dataset.

2.4.3 Without parallel data

While all the above mentioned methods rely on biligual word lexicons, most recent studies are aiming to eliminate the need for any parallel data at all. Smith et al. [54] has already made attempts for the alleviation of parallel data supervision by introducing **character-level information**, but the results were not on par with their supervised counterparts,

on the one hand, and on the other hand, these methods are strictly limited to pairs of languages sharing a common alphabet.

Conneau et al. [27] introduces an **unsupervised** way for aligning monolingual word embedding spaces between two languages **without using any parallel corpora**. Their experiments show that this method can be applied even for distant language pairs like English-Russian or English-Chinese.

On Dinu's standard word translation retrieval benchmark, using 200k vocabularies, their method reached **66.2%** accuracy on **English-Italian** while the best supervised approach is at **63.7%** (English-Italian, Precision @1).

Chapter 3

Proposed model

3.1 Multilingual data

In this section I briefly describe the data resources that I used. These involve the pre-trained embeddings I took for the experiments and the gold bilingual dictionaries I used for the evaluation.

3.1.1 English-Italian setup of Dinu

Dinu et al. [28] has constructed an English-Italian dictionary split into a train and a test set that are now being used as a benchmark data for evaluating word translation tasks.

Both train and test translation pairs are extracted from a dictionary built from Europarl, available at http://opus.lingfil.uu.se/ (Europarl, en-it) [57]. For the test set they used 1,500 English words split into 5 frequency bins, 300 randomly chosen in each bin. The bins are defined in terms of rank in the frequency-sorted lexicon: [1-5K], [5K-20K], [20K-50K], [50K-100K] and [100K-200K].

For the training translation pairs they also sampled by frequency, using the top 1K, 5K, 10K and 20K most frequent translation pairs from the Europarl dictionary sorted by English frequency. There is no overlap with test elements on the English side, however, when checking for overlaps with the 5K train data on the Italian side we found 113 Italian words that can be found both in the training and in the test sets. These overlaps can be sorted into the following categories:

- **Singular-plural correspondence:** in Italian when the last vowel of a substantive is accented the plural form is the same as the singular. For example *comunità* and *attività* in table 3.1.
- **Italian word is mistaken for English word:** the English translation is the same as the original Italian word. For example in the test set the Italian word *segni* is not translated and the same happens with *vecchi*. See table 3.2.
- **Different verb forms:** the same Italian word can be translated into different English verb tenses. For example *sostenere* in table 3.3.

- **Synonyms and homonyms:** one Italian word can be translated into several English words that might be synonyms or might not in case of homonyms. This phenomenon is actually fairly understandable and acceptable under all circumstances. See table 3.4.
- **Errors in the translation:** wrong translations. For example plural form Italian words *gatti* and *passeggeri* are translated both as the correct plural form and the incorrect singular form. See examples in table 3.5.

Italian	**English - train**	**English - test**
comunità	communities	community
attività	activities	activity

Table 3.1: *Singular-plural correspondence*

Italian	**English - train**	**English - test**
segni	signs	segni
vecchi	old	vecchi

Table 3.2: *Italian word is mistaken for English word*

Italian	**English - train**	**English - test**
sostenere	support	supporting

Table 3.3: *Different verb forms*

A summary of word counts can be seen in table 3.6.

Smith et al. [54] reported results on this English-Italian dataset both in English-Itanlian and Italian-English direction. They reproduced the methods of Mikolov [42], Faruqui [29] and Dinu. A summary of the English-Italian results can be found in table 3.7 and the Italian-English in table 3.8, respectively. All the methods turned to be more accurate when translating from English to Italian. This is not surprising at all, given the fact that many English words can be translated to either the male or female form of the Italian word.

3.1.2 The `fastText` embedding

Usual techniques for obtaining continuous word representation, i.e. word embeddings, is to represent each word of the vocabulary by a distinct vector, without parameter sharing. They ignore completely the morphology of words which is a significant limitation especially for agglutinating languages, e.g. Hungarian. In these languages new words are formed by stringing together morphemes which leads to large vocabularies and many rare words.

In 2017 the Facebook AI Research group proposed a new approach based on the skipgram model [41], but this time, contrary to the previously mentioned methods, parameter sharing was applied, since words are represented as a bag of character n-grams [26]. A vector representation is associated to each character n-gram, and the words are being represented

Italian	English - train	English - test
risposte	answers	responses
sufficiente	sufficient	enough

Table 3.4: *Synonyms and homonyms*

Italian	English - train	English - test	Explanation
gatti	cat	cats	it only means cats
passeggeri	passengers	passenger	it only means passengers

Table 3.5: *Errors in the translation*

Language	Set	# words
eng	train	3442
	test	1500
ita	train	4549
	test	1849

Table 3.6: *Statistics of word counts. The train set contains 5000 and the test set 1500 word pairs, respectively.*

Precision	@1	@5	@10
Mikolov et al. (2013b)	0.338	0.483	0.539
Faruqui et al. (2014)	0.361	0.527	0.581
Dinu et al. (2015)	0.385	0.564	0.639
Smith et al. (2017)	**0.431**	**0.607**	**0.664**

Table 3.7: *English to Italian results on Dinu's data published by Smith*

Precision	@1	@5	@10
Mikolov et al. (2013b)	0.249	0.410	0.474
Faruqui et al. (2014)	0.310	0.499	0.570
Dinu et al. (2015)	0.246	0.454	0.541
Smith et al. (2017)	**0.380**	**0.585**	**0.636**

Table 3.8: *Italian to English results on Dinu's data published by Smith*

as the sum of these representations. The method turned out to be fast and it allows us to compute word representations for words that did not appear in the training data. The model was evaluated both on word similarity and analogy tasks. The results show that this model outperforms Mikolov's `CBOW` and `skipgram` baseline systems that do not take into account subword information. It also does better than methods relying on morphological analysis.

The pre-trained word vectors for 294 languages, trained on Wikipedia using fastText are available on the following github link:
https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

3.1.3 Panlex

"Overcoming Language Barriers by Connecting Every Word in Every Language" that is the motto of PanLex, which is a nonprofit organiization with a mission to overcome language barriers to human rights, information, and opportunities [11]. They have been building a lexical database for over 10 years now, by transforming thousands of dictionaries into a single common structure. Thus, the PanLex database is a very powerful resource that I intend to use in this work as gold data.

The name PanLex is coming from of the words *panlingual* and *lexical*. The former one emphasizes the coverage of every single language, while the second one refers to its aim of translating words instead of sentences. Google Translate and other machine translation applications translate whole sentences and texts in up to a hundred major world languages, whereas PanLex focuses only on word translation, but in thousands of languages. The PanLex database is free for noncommercial use, and the translator itself can also be found online on the following website: https://apps.panlex.org/translator/.

The following link, https://panlex.org/source-list/, lists all the dictionary sources that are registered to the PanLex database. All of them have been acquired which means that either files have been downloaded, or books were purchased, scanned and adapted, but not all have been analyzed and incorporated into the PanLex database yet. It is possible to query specific sources, and for every entry we find the author, the publish year and the number of entries of a specific dictionary resource.

From a developer's view the PanLex project is focusing on the following tasks: *acquisition, assimilation, interface creation, research, infrastructure development*. Below I briefly summarize these tasks, and after that I emphasize the most important features of the data model and the database design of PanLex. To participate in the project you can apply as an itern or as a volunteer.

Acquisition Acquisition is the work of acquiring data for PanLex. The most important concept is the *expression* which refers to words and word-like phrases. Word-like phrases obviously contain more words, but the reason for that is that these phrases are necessarily have to be treated as whole, since word-by-word translation does not apply. What we can still consider as a word-like phrase is not quite clear so far, for example, one can debate whether "orange juice" counts as an *expression*, but "sweet potato" surely does. As for languages with various inflections (e.g. number, case etc.) it must be known that *expressions* only appear in their "lexeme" form. Translations can be unilingual (synonyms), bilingual, or multilingual, from which, keeping in mind the main purpose of the PanLex project, multilingual dictionaries are the most valuable. Furthermore, if an *expression* has translations into 20 different languages, then, instead of only 20 translations, in fact 210 ones are found.

The PanLex database has over a billion lexical translations if counting each pair of expressions between which there is at least one attested translation. Nonetheless, there is great disparity among languages. There are about 2,000 languages (recognized by the ISO 639-3 standard, http://www-01.sil.org/iso639-3/) with no expressions at all, and most of

the "threatened" or "endangered" languages are reflecting very "low-density". The priorities of acquisition are as follows:

- coverage: the more languages the better
- diversity: poorly documented language families have high priority
- laterality: still not contained dictionary pairs are more useful
- quality: qualified resources tend to be more useful
- efficiency: easy-to-convert data is preferred

Own translations are not accepted, they aim to collect already existing translations made by domain experts. The ownership of data is slightly controversial, but protection from public access and fair use of copyrighted works is guaranteed.

To facilitate the effort of the acquisition process various tools are provided. As for the project management system they use Wrike [18].

Assimilation Assimilation is the process of consulting the acquired data and to convert it to a form that complies with the standards of the PanLex database. The method of assimilation can vary greatly from one source to another, which might include using different types of programming languages, the application of different keyboards (e.g. Arabic, Chinese), and sometimes it is inevitable to understand the language itself in order to assimilate is correctly.

Interfaces The PanLex data can be accessed through various interfaces. Although interface development for human users is only a secondary purpose of PanLex, (since the primary one is to enrich the lexical database), there are still quite a few different available interfaces, including an online translator, standard HTTP API, custom search engines, and an integration with the NLTK [9] python library, called PanLex lite.

In addition, there are also some games that are being developed to make the exploration of PanLex expressions more interesting and to invite more people to use PanLex.

Research Current research fields, like translation inference, cross-language information retrieval, or cross-language human communication can make use of PanLex data. The PanLex itself also began as an applied research project. Current research topics regarding PanLex are focusing on the following areas:

- quality management: supervision of source data
- error detection: automated detection of errors in source data
- translation confidence: improvement of translation estimates that PanLex assigns to translations
- ontological enrichment: extension with language varieties, addition of conceptions, e.g. Morris Swadesh' "Swadesh list" [55]

- visualization: automated visualization of the PanLex database

Infrastructure Behind the database itself there is an extended infrastructure as well. Developments are carried out to improve the capabilities, design, performance and reliability.

Data model A detailed description of PanLex data model is available at https://dev.panlex.org/data-model/.
Here I would only like to explain the concept of a *language variety*. A *language variety* consists of a *language* and a three-digit *variety code*.

- *language*: is a three-letter ("alpha-3") code which complies with the ISO 639 protocol. For example in case of the English language the code is **eng** or as for Hungarian it is **hun** etc.
- *variety code*: is a three-digit code starting with **000**.
- *language variety* = *"language"-"variety code"* e.g. **eng-000** designates English. The most widely spoken variety of a language is often variety **000**, but there is no systematic rule for the assignment of variety codes. Variety codes include (but not limited to) regional variation and different writing systems. For example, **eng-004** designates American English.

In our experiments we always take the *language variety* with the smallest *variety code*.

Database design A detailed description of the PanLex database design is available at https://dev.panlex.org/database-design/.
What we need to extract from the PanLex database are expression pairs between two languages. For that we need to read the `langvar` table (language varieties, abbreviated as **lv**) and the `expr` table (expressions abbreviated as **ex**).

The code for extracting the expression pairs and creating a tsv file from them is available at:
https://github.com/Eszti/dipterv/blob/master/panlex/scripts/panlex/extract_tsv.py

TODO: what does score mean??

3.2 Description of our method

In this section I describe our proposed model in detail. I define the metrics that we are monitoring during training and evaluation processes, I introduce the equation we use for optimizing and I discuss the different configuration parameters of the system.

In a nutshell, in our work we are searching for linear mappings in form of translation matrices between pre-trained word embeddings through a universal vector space. We divide a gold dictionary data into training, development and test subsets. During training we

maximize the cosine similarity of word translations pairs represented as word vectors in a universal space. After having mapped the embeddings of two different languages into this universal space the cosine similarity of the actual translation pairs should be high, i.e. close to 1. At test time we evaluate our system with the precision metric used principally for word translation tasks.

3.2.1 Cosine similarity and precision

In word similarity tasks the extent to which the meanings of two words are similar is what we are interested in. **Cosine similarity** [3] is a measure of similarity between two non-zero vectors. It is calculated as the inner product of the two vectors divided by the product of the lengths of the two vectors, as shown in Equation **??**. Therefore, it is a space that measures the cosine of the angle of two vectors. In word similarity tasks the similarity of two words represented as word vectors are measured by the cosine similarity of the two vectors. The maximum value, +1, denotes complete similarity, 0 denotes neutrality, and the minimum value, -1, corresponds antonyms.

$$cosine\_similarity = cos(\theta) = \frac{\vec{a} \cdot \vec{b}}{||\vec{a}|| \cdot ||\vec{b}||} \tag{3.1}$$

In word translation tasks the performance is usually measured by the **precision** metric. In a word translation task there is always a look-up space which, for example, in our experiments corresponds to the most frequent 200k words of the given language. After translating a word we regard the N closes words to the translated vector in the look-up space. The metric **Precision @N** denotes the percentage of how many times the real translation of the word was found among the N closes words in the look-up space. Usual N values are 1, 5, and 10, respectively.

3.2.2 Equation to optimize

The objective of our method is to learn linear mappings in form of translation matrices that are obtained by maximizing the cosine similarity of gold word translation pairs in a universal space. Therefore, for each language one single translation matrix is searched that maps the language from its original vector space to a universal one.

Our objective is to bring close the translation pairs in the shared, universal space, therefore, our method is not only applicable for language pairs, but for any number of languages as well. The main advantage is that by introducing new languages the number of the learned parameters remains linear to the number of languages, since instead of learning pair-wise translation matrices for each language only one matrix is learned, the one that maps directly to this shared, universal space.

Let L be a set of languages, and TP a set of translation pairs where each entry is a tuple of two in form of (w_1, w_2) where w_1 is a word in L_1 language and w_2 is a word in L_2 language, and both L_1 and L_2 are in L. Then, let's consider the following equation, the Equation **??**:

$$\frac{1}{|TP|} \cdot \sum_{\substack{\forall L_1,L_2 \\ \in L}} \sum_{\substack{\forall (w_1,w_2) \in TP, \\ w_1 \in L_1, w_2 \in L_2}} cos\_sim(w_1 \cdot T_1, w_2 \cdot T_2) \tag{3.2}$$

where T_1 is the translation matrix that maps L_1, and T_2 which maps L_2 to the universal space, respectively. Since we normalize the equation with the number of translation pairs in the *TP* set, the optimal value of this function is 1. Off-the-shelf optimizers are programmed to find local minimum values. Therefore we must give our loss function (Equation **??**) multiplied by -1 to the optimizer, so that it would be a minimization task.

Note: if w_1 and w_2 values are normalized, as Xing et al. [59] suggested, the *cos_sim* reduces to the simple dot product of the translated vectors. In our experiments we are always working with normalized vectors. At test time we evaluate our system with the precision metric, more specifically with Precision @1, @5, and @10. The distance assigned to the word vectors in the look-up space is the *cosine_similarity*. The bigger this value is, the closer the two vectors are.

3.2.3 Configuration parameters

During the training process there are several configuration parameters that we adjusted using the development set. In this section I only recite them, the process of finding the optimal values and the results of experimenting with different setups are discussed in Chapter 4 in detail.

Generic parameters

The optimization process has several parameters that we can adjust. Below I enlist those ones that we experimented with:

- **optimizer:** the method to find the optimum value of the equation. Most common optimizers are: SGD (Stochastic Gradient Descent), Adagrad, Adadelta, Adam, Adamax [7]
- **epochs:** one epoch is the number of iterations after which we have seen each and every example of the training set exactly once. The more epoch we do, the more the system has learned.
- **batch size:** the number of examples we use in one iteration. Originally we differentiate three different types of SGD algorithms based on the batch size [14]:
  - **BGD (Batch Gradient Descent)** refers to the procedure when the batch size is equal to the number of all the training examples, i.e. one iteration is the same as one epoch.
  - **SGD (Stochastic Gradient Descent)** refers to the procedure when the batch size is equal to one, i.e. one epoch consists of as many iteration as the number of examples in the training set. It is often referred to as online learning, or noisy gradient descent.

  - **MBGD (Mini-batch Gradient Descent)** is a compromise between BGD and SGD, namely it means that the size of a batch can vary from 1 to N, where N denotes the number of all the training examples.

- **learning rate:** parameter which adjusts how fast is the learning process.
  - If the learning rate is **high** the learning process is faster, since we are heading towards the local minimum with bigger steps. The drawback is that this minimum can easily be missed. Since steps are big, what usually happens is that we keep jumping from one side of the minimum to the other side without ever reaching it, or sometimes we are even getting further from it.
  - If the learning rate is **low** the learning process is slower, because the steps taken towards the local minimum are smaller. With a smaller learning rate we tend to get closer to the real minimum point, although sometimes it takes so much that it does not worth waiting for it.
  - The task of adjusting the learning rate parameter is to find the **optimal** payoff between the time needed to run the experiments and the quality of the results that the experiments provide.

- **Batch size - learning rate relation:** Goyal et al.[34] studied the behaviour of different batch size and learning rate combinations, running their experiments on the ImageNet database [53]. As a rule of thumb they determined the following relation between these two parameters: if an experiment with a base batch size b and a base learning rate η terminates in time t, then if we increase the batch size by a factor of k, i.e. batch size $= b \cdot k$, then in order to keep the execution time at t we should apply $\eta \cdot k$ for the learning rate. In this case, in addition to the same execution times, the two learning processes are also having rougly the same learning curves, i.e. their loss functions over time are very similar.

Specific parameters

Besides the generic configuration parameters that have to be adjusted basically at all kind of machine learning or optimization tasks, we have some other, more task specific configuration parameters that we experimented with. These are the following ones:

- **SVD:** Smith et al. [54] suggested applying SVD (Singular Value Decomposition) to the transformation matrices, which turned out to be quite useful. Therefore, we also introduced a setting option whether to apply SVD or not.

- **SVD mode:** we introduce three different modes for experimenting with SVD. More specifically, they are called 0, 1, and 2. 0 means no SVD at all. 1 means doing an SVD regularly, i.e. on every n-th batch, and 2 means doing SVD only at the very beginning (after the first batch).

- **SVD frequency:** when applying SVD mode 1, this option corresponds n, i.e. the frequency how often an SVD will be applied on the translation matrices.

- **Embedding limit:** the number of words occuring in an embedding varies from language to language. In order to be able to evaluate the system for different languages equally we always read only the first n lines of the given word embeddings. This way the look-up space will have the same size for every language.

Parameters for evaluation

At test time we used different metrics for evaluation:

- **Precision:** the most important metric for evaluation is the precision. The system is capable of calculating any number of precisions, although in the end we figured that the most reasonable is to calculate those values only that are widely used by others as well, i.e. Precision @1, @5, and @10.
- **Loss:** at training time we optimize for the cosine similarity on the training set. It is also interesting to see what the results are on the test set.
- **Calculating small singular values:** small singular values of a translation matrix are indicators of dimension reduction of the translated space. Definitely we do not want that to happen, so it is also worth checking out the number of small singular values. The limit below which we consider a singular value as a small value is another configurable parameter.

3.3 Software architecture

In this section I describe briefly the software architecture and the used packages, tools and off-the-shelf codes I utilized during the implementation of the system.

3.3.1 Implementation

The implemented code is available as an open source project. The code can be found under my own github repository, in the following link:
https://github.com/Eszti/dipterv.
The proposed method is implemented in Python 3 [12] using the following python packages: numpy [10], matplotlib [8], sklearn [13], gensim [6], and tensorflow [15] (for more description see Section 3.3.2).

Config files

During development I made an effort to implement the system in a flexible and widely configurable way. This was very crucial since for parameter adjustment numerous experiments should be launched and the easiest way to cope with it is to leave the source code intact and modify only (most of the times manually) a human-readable config file.

Loading embedding files

Working with different languages implies working with different types of characters and character encodings. Handling all these different encodings has always been an issue for programmers. At the end of the 70s the American Standard Code for Information Interchange (a.k.a. **ASCII**) defined numeric codes for various characters, with the numeric values running from 0 to 127. For example, the lowercase letter 'a' is assigned 97 as its code value, but neither accented (like the Hungarian 'ü', 'ű' etc.) nor special non Latin characters could be represented. There were different encodings for different languages, such as KOI8 worked for Russian, or Latin1 for French, but problems always arose when you started using them together. So, therefore, **Unicode** was created to unify eventually this chaos. Unicode specification uses codes from 0 to 1,114,111 (0x10FFFF in base 16). Unicode characters are represented by code points which are integer values, usually denoted in base 16. To make them human readable they must be converted into a sequence of bytes. This process is called **encoding**. UTF-8 (Unicode Transformation Format using 8-bit numbers) is one of the most commonly used encodings. It uses two simple rules:

- If the code point is $<$ 128, it's represented by the corresponding byte value.
- If the code point is $>=$ 128, it's turned into a sequence of two, three, or four bytes, where each byte of the sequence is between 128 and 255.

Since Python 3.0, the language features a `str` type that contain Unicode characters and the default encoding for Python source code is UTF-8. This way we do not have to enforce manually the encoding-decoding processes. By putting a specially-formatted comments, we are able to use different encodings.

Working with multilingual embeddings always leads to encoding issues, therefore Pyhton 3 improvements came in handy. I represent embeddings as a **floating point matrix** with shape N*D (where N is the number of the words and D is the dimension of the embedding) and an **index2word** word list for assigning a word to each row of the matrix. In order to be able to load different formats of embeddings I created a base class for the common properties and I derived various classes that are responsible for handling different formats.

Obtaining the PanLex data

The code for extracting word pairs from the PanLex database (described in Section **??**) is a script created by my supervisor, Gábor Recski. I made some small modifications on it which uploaded to my github repository as a separate script. The script is available in the following link:
https://github.com/Eszti/dipterv/blob/master/panlex/scripts/panlex/extract$_t$*sv.py*

Training

The training process is carried out by TensorFlow. Matrices for word embeddings are initialized as placeholders, so in each iteration they are filled up with batch-size amount of

data. Validation is relatively expensive in time, so the frequency of how often to do it can also be set in the config file.

3.3.2 Brief description of the most important used packages

Numpy

Numpy is used for scientific computing with Python. It provides a powerful N-dimensional array object with sophisticated (broadcasting) functions. Also, it provides tools for integrating C/C++ and Fortran code. Besides, linear algebra, Fourier transform, and random number libraries are also very useful and easy to use [10].

TesorFlow

TensorFlow is an open source software library for high performance numerical computation across a variety of platforms (CPUs, GPUs, TPUs). It comes with strong support for machine learning and deep learning. Changing between different execution units like changing between CPU and GPU is transparent [15].

Chapter 4

Experiments

4.1 Baseline experimental setting

In this section I describe our baseline experimental setting that we used as a proof-of-concept and for parameter adjustment.

For our baseline system we used the `fasttext` embeddings (see 3.1.2) and Dinu's Enlgish-Itaian data (see 3.1.1). For parameter adjustment we split Dinu's training data into train and validation sets following their procedure, i.e. taking care of not having the same English word both in the training and in the validation set as well. Note, that it does not apply for the Italian words, where we do have a significant overlap (80 words), see Table 4.1. For overlaps between original train and test data see Table 4.2.

# words English	train	3098
# words Italian		4129
# words English	valid	344
# words Italian		499
overlap English		0
overlap Italian		80

Table 4.1: *Splitting training data into training and validation*

# words English	train	3442
# words Italian		4549
# words English	test	1500
# words Italian		1849
overlap English		0
overlap Italian		113

Table 4.2: *Original train and validation data*

Following, we trained our system on the training data with the proposed procedure described in 3.2. For **optimizer** we used Adagrad [7] since it is said to be TODO: why??.

For **evaluation** we take the 200k most frequent words of the embeddings and use them as the look-up space for calculating Precision @1, @5, and @10. In all cases we calculate both English-Italian and Italian-English precision scores. Besides, we also check the average

cosine similarity value of the validation set. Both precision and similarity values are calculated in the **universal space**, during training and validation as well. Gold dictionaries were constructed from the input data files themselves. Following Dinu, we considered any words appearing in the dictionary a valid translation (e.g. synonyms, male-female forms etc.) [28].

4.1.1 Adjusting basic parameters

With the above described experimental setting we searched for the best learning rate and batch size setting. First, we found the most appropriate learning rate using a default, fixed batch size (64), and then we used this learning rate for the batch size experiments. In all cases we trained for 10k epochs, and we applied an initial SVD (SVD mode 2), described in more detail in Section 4.1.2. Over the 10k epochs we ran an evaluation on the validation set at every 1000th epoch. In the tables the maximum precision values are shown which, in most of the cases, are not from the last epoch. We did want to see the curve to break down, to reach over-fitting, so that we could be convinced that the system was trained long enough.

Learning rate

For learning rate experiments we fixed the value of batch size at 64 and ran various experiments with the following learning rates: 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 3 (suggested by Andrew Ng in the Stanford Machine Learning Coursera course [14]). Table 4.3 summarizes the experiments. As we can see the best results occur when the learning rate is 0.1, so later, at the batch rate experiments we fixed the learning rate to 0.1.

LR	**cos_sim**	**English - Italian Precision**			**Italian - English Precision**			**Time**
		@1	**@5**	**@10**	**@1**	**@5**	**@10**	
0.001	0.988743	0.1831	0.1831	0.3721	0.1667	0.2851	0.3494	∼1:35
0.003	0.995905	0.3401	0.5058	0.5669	0.3032	0.4799	0.5462	∼1:20
0.01	0.998957	0.4651	0.6366	0.6802	0.4036	0.6185	0.6586	∼1:25
0.03	0.999824	0.5262	0.7006	0.7645	0.4438	0.6506	0.6988	∼1:15
0.1	0.999994	**0.5407**	**0.7297**	**0.7645**	**0.4618**	**0.6546**	0.6948	∼1:20
0.3	1.000000	0.5407	0.7151	0.7645	0.4478	0.6526	**0.7028**	∼1:35
1	1.000000	0.4535	0.6483	0.6977	0.3554	0.5542	0.6265	∼1:35
3	1.000000	0.0698	0.1599	0.1890	0.0462	0.0462	0.1586	∼1:45

Table 4.3: *Learning rate experiments. "LR" stands for learning rate, and "cos_sim" denotes the average cosine similarity of the training set. Time is shown in h:mm format.*

Batch size

Having fixed the learning rate to 0.1 we ran various experiments with the same experimental setting using the following batch sizes: 16, 32, 64, 128, 256. Table 4.4 summarizes the results. Since the batch size of 64 provides most of the times the best results on the validation set,

for future experiments we set the learning rate to **0.1** and the batch size to **64**, which, by the way, happened to be our first intuition.

BS	cos_sim	English - Italian Precision			Italian - English Precision			Time
		@1	@5	@10	@1	@5	@10	
16	1.000000	0.5320	0.7209	0.7616	0.4418	0.6446	0.7008	∼3:20
32	1.000000	0.5203	0.7064	0.7558	0.4398	0.6446	0.6948	∼2:00
64	0.999994	**0.5465**	0.7209	**0.7878**	**0.4578**	**0.6627**	0.7068	∼1:10
128	0.999946	0.5407	**0.7267**	0.7645	0.4458	0.6586	**0.7129**	∼0:55
256	0.999949	0.5320	0.7093	0.7645	0.4398	0.6627	0.7088	∼1:25

Table 4.4: *Batch size experiments. "BS" stands for batch size, and "cos_sim" denotes the average cosine similarity of the training set. Time is shown in h:mm format.*

Conclusions

Figure 4.1 shows the learning curve of the experiment with learning rate = 0.1 and batch size = 64. The red line shows the average cosine similarity on the training set, and the green line on the validation set, respectively. Validation was done only 10 times over the 10k epochs, so compared to the training curve the validation curve is obviously very steep in the beginning.

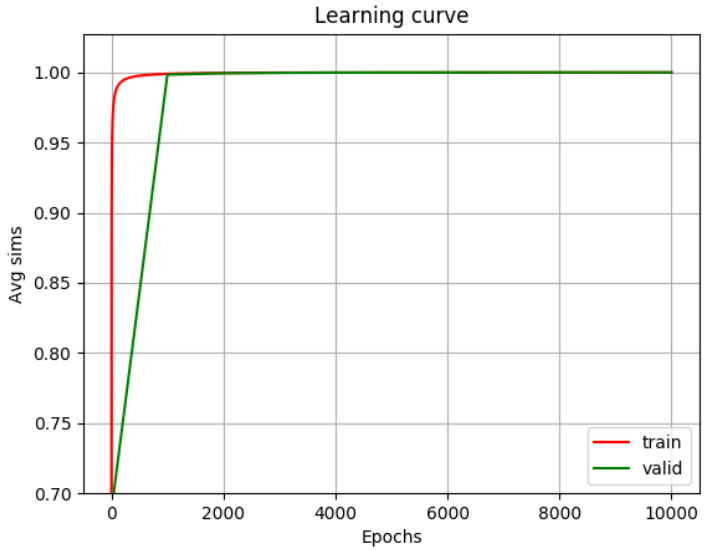

Figure 4.1: *Learning curve of experimenting with learning rate = 0.1, batch size = 64.*

On Figure 4.2 we can see the precision curves of English-Italian, while on Figure 4.3 the precision curves of Italian-English word translation of the same experiment. We can observe that as the average cosine similarity is getting higher, the precision is growing as well. After a certain point, however, the precision curves start to decrease, since we are facing the classical over-fitting problem.

These experiments also serve as a proof-of-concept for our method. By optimizing on cosine similarity, once the translation matrices are learned we want to be able to use

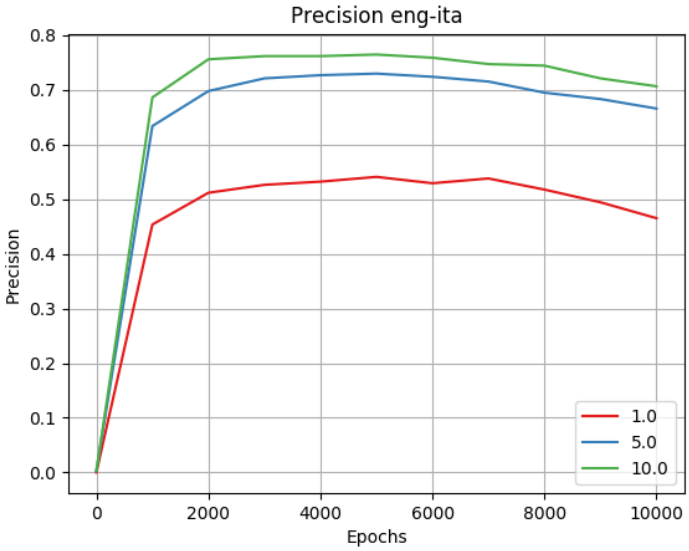

Figure 4.2: *Precision curve eng-ita when experimenting with learning rate = 0.1, batch size = 64. The red curve is Precision @1, the blue is @5, and the green is @10.*

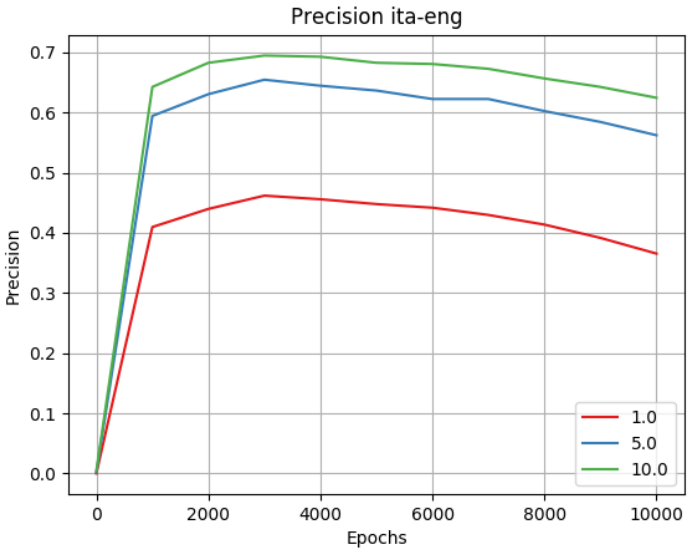

Figure 4.3: *Precision curve ita-eng when experimenting with learning rate = 0.1, batch size = 64. The red curve is Precision @1, the blue is @5, and the green is @10.*

our method for various multilingual applications, such as for word translation tasks. The results of the experiments above show that there is a clear correlation between similarity and precision values.

4.1.2 Experimenting with SVD

Previous works suggested restricting the transformation matrix to an orthogonal one (Smith et al. [54], Conneau et al. [27]). Based on their work we also studied the behaviour

of applying SVD on the translation matrix. This feature is configurable and is denoted to the config parameter, `SVD_mode`. We inspected 3 different settings with the train and validation datasets described in Section 4.1:

- **0** not using SVD at all
- **1** using SVD after every n-th epoch
- **2** using SVD only once, at the beginning

From a random transformation matrix T we obtain the orthogonal one, T' by applying SVD the following way:

$$S, U, V = SVD(T) \tag{4.1}$$

$$T' = U \cdot V \tag{4.2}$$

Base on the previous findings, in these experiments we set the learning rate to 0.1 and the batch size to 64. Each time we ran 200 epochs, and evaluated on every 10th epoch.

`SVD_mode` = **0**

This experiment is carried out without applying any SVD. We initialize the tranlation matrices with random numbers and let the system learn by itself.

On Figure 4.4 we can see that the similarity values are monotone increasing, the system does learn. But the learning process is relatively slow since even after 200 epochs the similarity score is still quite low (we want to reach 1.0).

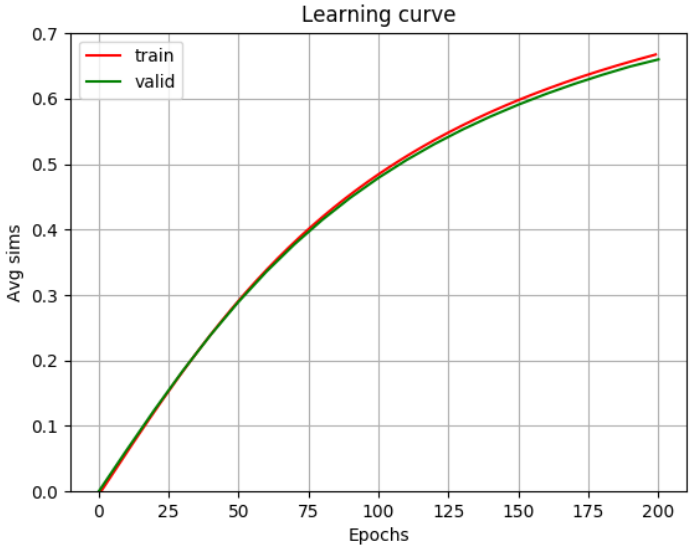

Figure 4.4: *Learning curve of experimenting with svd_ mode = 0.*

`SVD_mode` = **1**

This experiment is carried out with applying SVD various times over the whole learning process. Just like the other two cases we trained the system for 200 epochs, and we made and SVD on every 50th epoch, i.e. 4 times altogether.

On Figure 4.5 we can see how the learning curve breaks down every time after applying an SVD on the translation matrices, and, also, that how fast it is back once again to the previous high similarity values. Besides, if we compare the similarity values to those without SVD from the previous experiment, we can see that this time, even right at the beginning, the average cosine similarity score is already way higher than it was after 200 epochs without SVD. Applying SVD on the transformation matrices seems to accelerate the learning process significantly.

We can also see that SVD-to-SVD fractions of the learning curve seem to have exactly the same trajectory, regardless of the number of previous epochs done. As a result, we can conclude that it is not worth applying SVD repeatedly. For this reason we introduced `svd_mode` = 2, which stands for the setting when SVD is applied only once all over the whole training process, it is applied at the beginning, right after the initialization of the translation matrices.

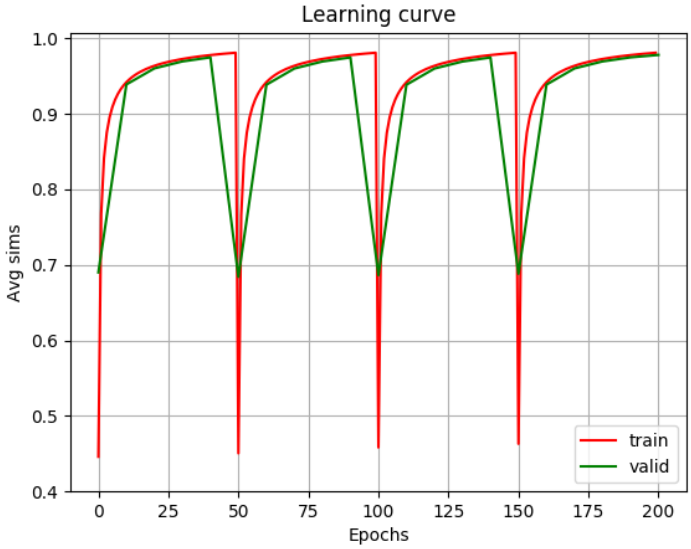

Figure 4.5: *Learning curve of experimenting with svd_ mode = 1.*

`SVD_mode` = **2**

This experiment is carried out with applying SVD only once, at the very beginning. That means, basically, that instead of a random initial transformation matrix, we already start with an orthogonal one.

On Figure 4.6 we observe that the learning curve is monotone increasing, and thanks to the initial SVD it gets fairly high right at the beginning.

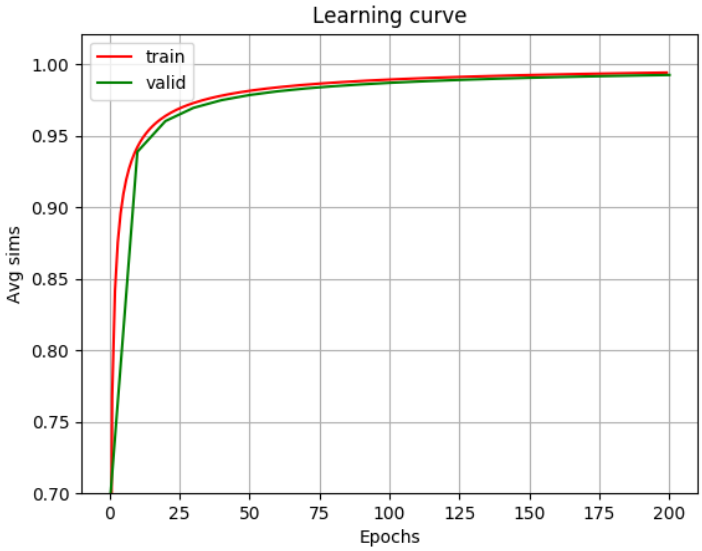

Figure 4.6: *Learning curve of experimenting with svd_mode = 2.*

Dimensionality loss in universal space

Still increasing similarity scores in parallel with decreasing precision is the typical pattern of over-fitting in machine learning applications. Although we do not use classical machine learning, merely a vanilla SGD for optimization, this phenomenon can still occur. One possible explanation is the reduction of dimensionality in the universal space, which also implies information loss that can lead to decreased precision values. One indicator of this problem is when the number of small singular values of the translation matrix is high. In order to monitor this we studied the number of singular values less than 0.1 by different number of epochs. The results can be seen in Table 4.5. We can observe that as the average similarity is monotone increasing (both by training and validation), the number of small singular values of the translation matrices is increasing as well.

These results were obtained from the same experimental setting that we can see in Figures 4.1, 4.2, and 4.3 (learning_rate = 0.1, batch_size = 64, SVD_mode = 2). The singular values of a matrix can be found in the S matrix after performing SVD.

Conclusion

Based upon previous works we also implemented a feature of performing SVD. We tried 3 different settings; not using SVD, using it at every nth epoch, and using it only once. We observed that SVD significantly accelerates the convergence, and we concluded that the most effective way is performing SVD only once, right at the beginning, so that the initial translation matrix is orthogonal. We also observed that there is an obvious correlation between the increase of small singular values and the decrease of precision. This is due to dimensionality reduction in the universal space. The top system is the optimum, where the average cosine similarity is already high enough, but the number of small singular values are not yet. In case of the experiment shown in Table 4.5 the optimum is around 2000-3000

Epoch	# <**0.1 (eng)**	# <**0.1 (ita)**	**train**	**valid**
0	0	0	0.447719	0.687022
1000	24	27	0.998958	0.998392
2000	76	68	0.999627	0.999369
3000	120	113	0.999823	0.999684
4000	157	153	0.999905	0.999824
5000	190	188	0.999946	0.999896
6000	215	215	0.999967	0.999936
7000	237	237	0.999979	0.999959
8000	255	257	0.999987	0.999974
9000	258	270	0.999991	0.999983
10000	278	280	0.999994	0.999988

Table 4.5: *Monitoring dimensionality loss in universal space*

eng words	train	3442	test	1500
not found		0		97
ita words		4548		1849
not found		1		156
word pairs		5000		1869
found		4999		1640

Table 4.6: *Dinu's data statistic with* `fasttext` *embedding*

epochs, as it can be seen in Figure 4.2 and 4.3.

4.2 Dinu's experimental setting and our baseline system

With our best setting so far we ran one experiment with Dinu's original data setting described in 3.1.1 using first the `fasttext` embedding described in 3.1.2 and then Dinu's original embedding [28]. A summary of the results and a comparison with previous works can be seen in Table 4.7 and 4.8.

4.2.1 `fasttext`

In this experiment we trained on 4999 word pairs, and we tested on 1640 word pairs. Originally Dinu's data has 5000 word pairs in the training set and 1869 word pairs in the test set. The decreased number is because some words are not found in the `fasttext` embedding. Table 4.6 summarizes this data information.

Figure 4.7 shows eng-ita Precision scores, while Figure 4.8 the ita-eng ones. Unsurprisingly, English-Italian direction performs better, given that some English words in the test set can translate to either the male or female form. Smith et al. [54] came to the same conclusions.

Table 4.7 presents the results in English-Italian, and Table 4.8 in Italian-English direction. Our results are worse than Smith's but they are comparable or even better than previous results.

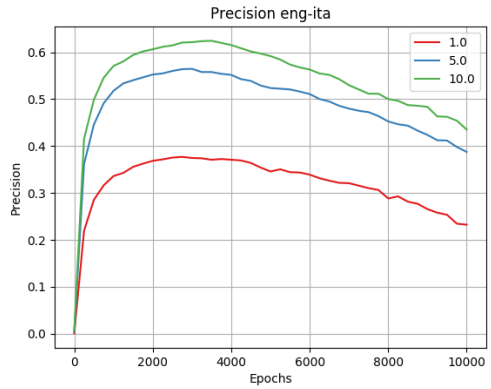

Figure 4.7: *Precision curve eng-ita of our method on Dinu's data using* `fasttext` *embedding.*

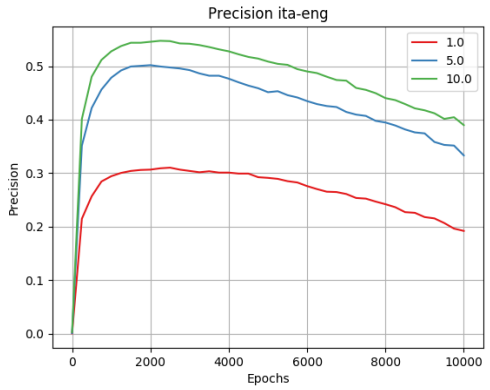

Figure 4.8: *Precision curve ita-eng of our method on Dinu's data using* `fasttext` *embedding.*

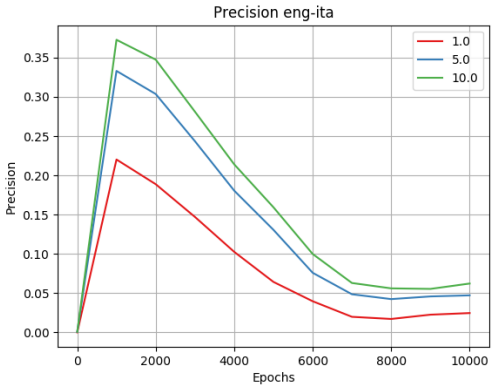

Figure 4.9: *Precision curve eng-ita of our method on Dinu's data using Dinu's embedding.*

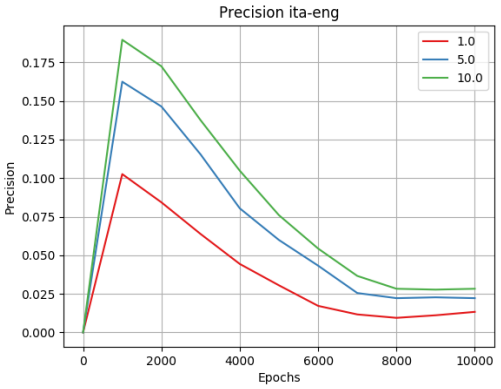

Figure 4.10: *Precision curve ita-eng of our method on Dinu's data using Dinu's embedding.*

4.2.2 Dinu's word vectors

Next we ran the system with Dinu's embedding as well. These word vectors were trained with `word2vec` and then the 200k most common words in both the English and Italian corpora were extracted. The English word vectors were trained on the WackyPedia/ukWaC and BNC corpora, while the Italian word vectors were trained on the WackyPedia/itWaC corpus. The data is available at: http://clic.cimec.unitn.it/ georgiana.dinu/down/.

This time we trained the system on 4912 word pairs (out of 5000) and tested on 1823 word pairs (out of 1869). The reason for this defect is the same as in the previous case, it is due to incomplete word embedding coverage.

Figure 4.9 shows eng-ita Precision scores, while Figure 4.10 the ita-eng ones. Once again English-Italian direction performs better than Italian-English direction, as it is expected.

Table 4.7 presents the results in English-Italian, and Table 4.8 in Italian-English direction. Our results are way behind of Smith's and they are also worse than our previous results with the `fasttext` embeddings.

Eng-Ita	**@1**	**@5**	**@10**
Mikolov et al.	0.338	0.483	0.539
Faruqui et al.	0.361	0.527	0.581
Dinu et al.	0.385	0.564	0.639
Smith et al.	**0.431**	**0.607**	**0.664**
Our method with fasttext	*0.3770*	*0.5647*	*0.6245*
Our method with Dinu's data	*0.2202*	*0.3331*	*0.3728*

Table 4.7: *Comparing our English-Italian results on Dinu's data with others.*

Ita-Eng	**@1**	**@5**	**@10**
Mikolov et al.	0.249	0.410	0.474
Faruqui et al.	0.310	0.499	0.570
Dinu et al.	0.246	0.454	0.541
Smith et al.	**0.380**	**0.585**	**0.636**
Our method with fasttext	*0.3103*	*0.5018*	*0.5474*
Our method with Dinu's data	*0.1026*	*0.1625*	*0.1897*

Table 4.8: *Comparing our Italian-English results on Dinu's data with others.*

4.3 Panlex experiments

In this section I write about the experiments carried out on the PanLex database. First, I summarize the results of our data analysis, then, I describe the experiments in detail. Finally, I report the obtained results.

4.3.1 Data inspection

In this section I present summary tables about the analysis of the PanLex database. In the PanLex database *translations* are scored according to the reliability of the source they are coming from. Since a *translation* might be found in different PanLex sources as well, one translation pair may also appear multiple times in the databse after joining the tables, and sometimes even with different scores. As a rule of thumb, when extracting the needed data from the PanLex database, first, we sort the entries according to their reliability score into a descending order, and then, we drop the duplicates. As a result each translation pair is represented with its highest score that it can be found with. Extraction of translation pairs were carried out with the following code:
https://github.com/Eszti/dipterv/blob/master/panlex/scripts/panlex/extract_tsv.py.

Table 4.9 (English-Italiean) summarizes the analysis results. Scores are going form 1 to 9, with 9 denoting the most reliable sources and 1 the least ones. In the second column we can see how many entries are found with a certain score value. In the third column we can observe the number of entries after having filtered the entries by keeping only those ones for which a word vector was found in the `fasttext` embedding. The last column adds up all the valid entries above a certain score. (By valid entries we mean those to which a word vector can be assigned.)

score	# wp == score	# wp == score, filtered	# wp >= score
9	1389	66	66
8	4265	514	580
7	163701	69043	69623
6	1085	67	69690
5	79419	26478	96168
4	6045	2836	99004
3	272276	47477	146481
2	126837	36182	182663
1	6893	4938	187601

Table 4.9: *Summary of English-Italian PanLex data inspection*

4.3.2 Training on PanLex

In this section I describe the experiments I ran using only the PanLex data both for training and evaluation as well. First I used a one order bigger set of data compared to Dinu's data 3.1.1, and then I ran experiments with the same size of data as Dinu's, but extracted from the PanLex database.

Experimenting with a bigger dataset

Considering Table 4.9 we concluded that using only the words with greater or equal to a score of 8 would result in a rather small dataset, since there are only 580 word pairs meeting this requirement. Thus, for all the PanLex experiments we took the word pairs with at least a score of 7. There are 69623 such word pairs in the PanLex dataset. First we split this set into a training and a test set (70% - 30%), following the procedure of Dinu, i.e. taking care of not having the same English words in both sets. Next, we split the training set into training and validation sets (90% - 10%). Table 4.10 shows a summary about the number of word pairs in each dataset.

sum	69623	train (70 %)	48472	test (30 %)	21151
train sum	48472	train (90 %)	43383	valid (10 %)	5089

Table 4.10: *PanLex dataset splits (score >= 7).*

In our first experiment we experimented with the training-validation set (43383 - 5089). We ran the training for 500 epochs, using 0.1 for learning rate, and 64 for batch size, and we did one SVD at the beginning, as it turned out to be the best setting for Dinu, described in 4.1.1 Section.

After that, we ran an evaluation both on our PanLex test set (21151 word pairs), and on Dinu's test set (1869 word pairs). The results can be seen in Table 4.11.

Sadly the results are rather disappointing. The system did not succeed in learning the transformations correctly, its performance is more than one order worse than our performance using Dinu's data for training as well. See previous results in Table 4.7 and 4.8.

	eng - ita			ita - eng			
	@1	**@5**	**@10**	**@1**	**@5**	**@10**	**# word pairs**
training	0.0328	0.0705	0.0911	0.0126	0.0324	0.0445	43383 - 5089
test on PanLex	0.0285	0.0601	0.0830	0.0177	0.0427	0.0569	21151
test on Dinu	0.0197	0.0379	0.0484	0.0228	0.0493	0.0616	1869

Table 4.11: *PanLex experiments trained on the big dataset*

Experimenting with a smaller dataset

Besides, we also created a smaller dataset for training and testing out of word pairs with greater or equal to 7 scores. These datasets both contain 5000-5000 word pairs (both for training and testing), just like Dinu's data does (Dinu has 5000 word pairs in the training set, in the test set there are only 1869 word pairs). The training word pairs were extracted from the original 70 % train split of the whole data, and the test word pairs from the original 30 % test split of the whole data. (First row of Table 4.10.) This time we extracted the words in a way that all English and Italian words are appearing exactly once at the set. (That is, neither both feminine and masculine, nor both singular and plural forms were allowed.)

We tried out this dataset with different learning rates 4.12 and batch sizes 4.13, with doing an SVD only once, at the first epoch. Results are not promising at all, but we came to a similar conclusion, like at Dinu's data. The best learning rate turned out to be clearly 0.1 in English-Italian direction, while in Italian-English direction, 0.3 was slightly better. As for the batch size, for English-Italian 64 is the most appropriate choice, while for Italian-English a little bit smaller batch size, 32, gave better results. Yet, in future settings we kept 0.1 for learning rate, and 64 for batch size.

	eng - ita			ita - eng		
lr	**@1**	**@5**	**@10**	**@1**	**@5**	**@10**
0.03	0.0294	0.0661	0.0872	0.0119	0.0283	0.0416
0.1	**0.0361**	**0.0750**	**0.0977**	0.0121	**0.0324**	0.0426
0.3	0.0278	0.0694	0.0938	**0.0128**	0.0312	**0.0450**

Table 4.12: *Learning rate experiments with the PanLex data.*

	eng - ita			ita - eng		
bs	**@1**	**@5**	**@10**	**@1**	**@5**	**@10**
32	0.0300	0.0694	0.0966	**0.0138**	**0.0332**	**0.0433**
64	**0.0361**	**0.0750**	**0.0977**	0.0121	0.0324	0.0426
128	0.0278	0.0633	0.0883	0.0143	0.0315	0.0428

Table 4.13: *Batch size experiments with the PanLex data.*

Investigating the problem, we saw that the main problem why the system is not giving good-enough precision scores on the validation set, is because is simply projects every single vector pretty close to every other vector in the universal space. Which is understandable, since the trivial solution of our Equation **??** is to set the translation matrices equal to a zero

	eng - ita			ita - eng		
SVD_freq	**@1**	**@5**	**@10**	**@1**	**@5**	**@10**
1	0.0272	0.0533	0.0788	0.0087	0.0211	0.0298
10	0.0228	0.0572	0.0805	0.0061	0.0179	0.0240
50	0.0328	**0.0783**	**0.1011**	0.0126	0.0286	0.0414
100	**0.0361**	0.0766	0.0994	**0.0140**	0.0327	**0.0431**
200	0.0300	0.0761	0.1005	**0.0140**	**0.0341**	**0.0431**

Table 4.14: *PanLex experiments with different SVD frequencies.*

matrix. To overcome this problem we introduced the SVD procedure that we apply on the translation matrices, and that guarantees that the translation matrix remains orthogonal, thus the dimension of the mapped spaces would not collapse. We tried applying the SVD with different frequencies 4.14. Some results could sightly overtake the previous results from Table 4.11 but they are still at the same order.

4.3.3 Training on Dinu, testing on PanLex

Following we wondered how the system trained on Dinu's training data would perform on the PanLex test set. On Figures 4.7 and 4.8 we see that the curves reach their maximum around 2000 epochs or maybe a little bit later. Since during that training we saved the translation matrices on every 1000th epoch, this time we ran evaluations using the translation matrices we obtained after 2000, 3000, and 4000 epochs, and as for evaluation data we took our small PanLex test set, containing 5000 word pairs.

	eng - ita			ita - eng		
epochs	**@1**	**@5**	**@10**	**@1**	**@5**	**@10**
2000	**0.1782**	**0.3124**	**0.3582**	**0.1712**	**0.2858**	**0.3248**
3000	0.1778	0.3104	0.3534	0.1670	0.2756	0.3178
4000	0.1738	0.2960	0.3396	0.1586	0.2638	0.3032

Table 4.15: *Evaluation results of transformation matrices trained on Dinu, tested on PanLex.*

Table 4.15 shows the test results. Best results were obtained by using the translation matrices after 2000 epochs of training. We can see that these numbers are one order greater than the previous ones, although they are still significantly worse than our numbers obtained by using Dinu's test set for evaluation. This can be a proof that the PanLex data has a lower quality, than Dinu's data, or that it is not appropriate for this kind of experimenting. What is also remarkable is that this time English-Italian, and Italian-English results are more close to one another, than in previous cases. The English-Italian, generally, is still better, but for example at Precision @1 the Italian-English is really close behind. This difference may be due to the balanced test set that does not contain words neither in English nor in Italian more than one time.

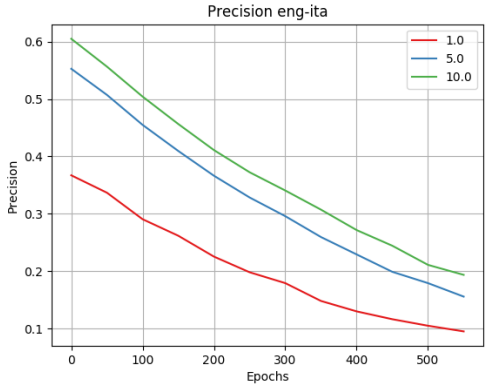

Figure 4.11: *Precision curve eng-ita when continuing with the big PanLex dataset.*

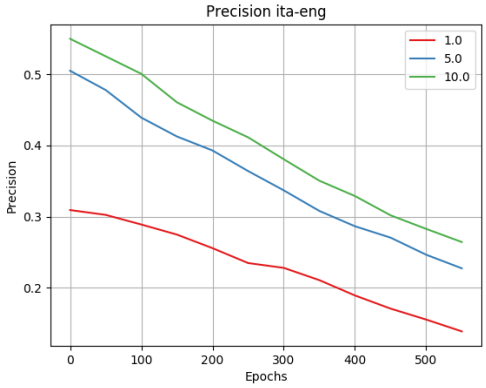

Figure 4.12: *Precision curve ita-eng when continuing with the big PanLex dataset.*

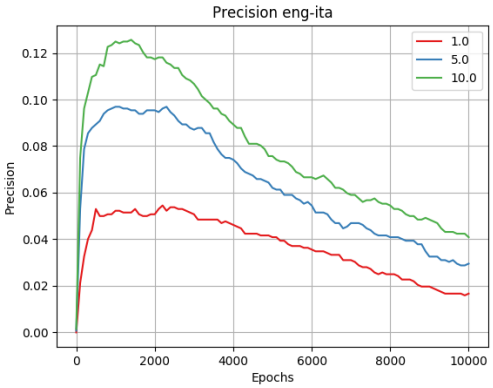

Figure 4.13: *Precision curve eng-ita when continuing with the small PanLex dataset and SVD.*

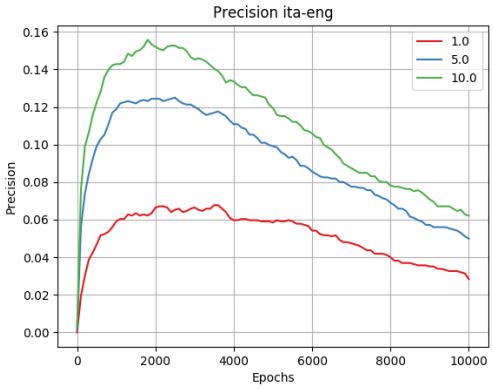

Figure 4.14: *Precision curve ita-eng when continuing with the small PanLex dataset and SVD.*

4.4 Continuing the baseline system with PanLex data

At last, we tried continuing the training process of our baseline system (trained on Dinu's data) with the PanLex data. We tried both with our bigger and with our smaller PanLex dataset as well. In the former case, the appr. 50k (noisy) word pairs quickly spoiled both in English-Italian and in Italian-English directions the former acceptable results as we can see in Figures 4.11 and 4.12. In this experiment no SVD was applied.

In the latter case, when we used the smaller, 5k dataset, we experimented both with doing an SVD at the beginning, and without doing an SVD at all. When applying an SVD we need significantly more epochs, than in the other case. In Figures 4.13 and 4.14 we can see the precision curves of experiments with applying an SVD, whereas in Figures 4.15 and 4.16 without applying it.

Table 4.16 summarizes the best obtained results after continuing the training process of Dinu's data with PanLex data, and it also compares these results with our previous ones. We can see that applying SVD barely manages to learn, but if we do not apply any SVD just merely let it run with the PanLex data, although English-Italian results are decreasing,

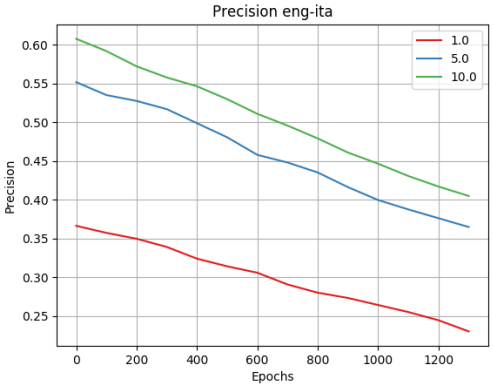

Figure 4.15: *Precision curve eng-ita when continuing with the small PanLex dataset and without SVD.*

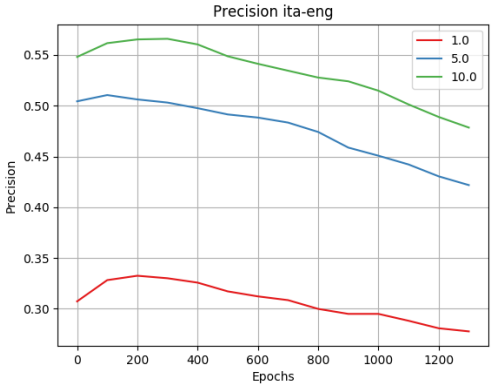

Figure 4.16: *Precision curve ita-eng when continuing with the small PanLex dataset and without SVD.*

Italian-English results are surprisingly increasing a little bit in the beginning.

	eng - ita			**ita - eng**		
	@1	**@5**	**@10**	**@1**	**@5**	**@10**
our best on Dinu	**0.3770**	**0.5647**	**0.6245**	0.3103	0.5018	0.5474
with SVD	0.0545	0.0969	0.1257	0.0677	0.1250	0.1558
without SVD	0.3664	0.5519	0.6079	**0.3325**	**0.5105**	**0.5659**

Table 4.16: *Results of continuing previously trained-by-Dinu's-data matrices with PanLex data.*

Chapter 5

Conclusions and future work

5.1 Summarizing the contributions of the thesis

5.2 Future word

Köszönetnyilvánítás

Ez nem kötelezõ, akár törölhetõ is. Ha a szerzõ szükségét érzi, itt lehet köszönetet nyilvánítani azoknak, akik hozzájárultak munkájukkal ahhoz, hogy a hallgató a szakdolgozatban vagy diplomamunkában leírt feladatokat sikeresen elvégezze. A konzulensnek való köszönetnyilvánítás sem kötelezõ, a konzulensnek hivatalosan is dolga, hogy a hallgatót konzultálja.

List of Figures

2.1 Network architecture proposed by Bengio et al.[25] 11
2.2 Bag-of-words neural networks suggested by Mikolov et al.[43] 12
2.3 Translating MOON and SUN through polysemous words. 14
2.4 Making links between English concepts through eliminating the internal nodes. 14
2.5 Levels of representation in Machine Translation. $\pi \rightarrow \sigma : tree - to - string; \sigma \rightarrow \sigma : string - to - string; \sigma \rightarrow \pi : string - to - tree$. 15

4.1 Learning curve of experimenting with learning rate = 0.1, batch size = 64. . 35
4.2 Precision curve eng-ita when experimenting with learning rate = 0.1, batch size = 64. The red curve is Precision @1, the blue is @5, and the green is @10. 36
4.3 Precision curve ita-eng when experimenting with learning rate = 0.1, batch size = 64. The red curve is Precision @1, the blue is @5, and the green is @10. 36
4.4 Learning curve of experimenting with svd_mode = 0. 37
4.5 Learning curve of experimenting with svd_mode = 1. 38
4.6 Learning curve of experimenting with svd_mode = 2. 39
4.7 Precision curve eng-ita of our method on Dinu's data using `fasttext` embedding. 41
4.8 Precision curve ita-eng of our method on Dinu's data using `fasttext` embedding. 41
4.9 Precision curve eng-ita of our method on Dinu's data using Dinu's embedding. 41
4.10 Precision curve ita-eng of our method on Dinu's data using Dinu's embedding. 41
4.11 Precision curve eng-ita when continuing with the big PanLex dataset. . . . 46
4.12 Precision curve ita-eng when continuing with the big PanLex dataset. . . . 46
4.13 Precision curve eng-ita when continuing with the small PanLex dataset and SVD. 46
4.14 Precision curve ita-eng when continuing with the small PanLex dataset and SVD. 46
4.15 Precision curve eng-ita when continuing with the small PanLex dataset and without SVD. 47
4.16 Precision curve ita-eng when continuing with the small PanLex dataset and without SVD. 47

List of Tables

3.1 Singular-plural correspondence . . . 22
3.2 Italian word is mistaken for English word . . . 22
3.3 Different verb forms . . . 22
3.4 Synonyms and homonyms . . . 23
3.5 Errors in the translation . . . 23
3.6 Statistics of word counts. The train set contains 5000 and the test set 1500 word pairs, respectively. . . . 23
3.7 English to Italian results on Dinu's data published by Smith . . . 23
3.8 Italian to English results on Dinu's data published by Smith . . . 23

4.1 Splitting training data into training and validation . . . 33
4.2 Original train and validation data . . . 33
4.3 Learning rate experiments. "LR" stands for learning rate, and "cos_sim" denotes the average cosine similarity of the training set. Time is shown in h:mm format. . . . 34
4.4 Batch size experiments. "BS" stands for batch size, and "cos_sim" denotes the average cosine similarity of the training set. Time is shown in h:mm format. . . . 35
4.5 Monitoring dimensionality loss in universal space . . . 40
4.6 Dinu's data statistic with `fasttext` embedding . . . 40
4.7 Comparing our English-Italian results on Dinu's data with others. . . . 42
4.8 Comparing our Italian-English results on Dinu's data with others. . . . 42
4.9 Summary of English-Italian PanLex data inspection . . . 43
4.10 PanLex dataset splits (score >= 7). . . . 43
4.11 PanLex experiments trained on the big dataset . . . 44
4.12 Learning rate experiments with the PanLex data. . . . 44
4.13 Batch size experiments with the PanLex data. . . . 44
4.14 PanLex experiments with different SVD frequencies. . . . 45
4.15 Evaluation results of transformation matrices trained on Dinu, tested on PanLex. . . . 45
4.16 Results of continuing previously trained-by-Dinu's-data matrices with PanLex data. . . . 47

Acronyms

NLP Natural Language Processing. 6

Bibliography

[1] https://www.youtube.com/playlist?list=PL3FW7Lu3i5Jsnh1rnUwq$_T cylNr7EkRe$6.

[2] https://www.economist.com/technology-quarterly/2017-05-01/language.

[3] Cosine similarity. https://en.wikipedia.org/wiki/Cosine$_s imilarity$.

[4] Facebook: M suggestions. https://newsroom.fb.com/news/2017/04/m-now-offers-suggestions-to-make-your-messenger-experience-more-useful-seamless-and-delightful/.

[5] Facebook: Recommendations. https://code.facebook.com/posts/550719898617409/under-the-hood-multilingual-embeddings/.

[6] Gensim. https://pypi.python.org/pypi/gensim.

[7] Keras optimizers. https://keras.io/optimizers/.

[8] matplotlib. https://matplotlib.org/.

[9] Nltk. http://www.nltk.org/.

[10] Numpy. http://www.numpy.org/.

[11] Panlex. https://panlex.org/.

[12] Python 3. https://www.python.org/download/releases/3.0/.

[13] Skikit-learn. http://scikit-learn.org/stable/.

[14] Stanford: Machine learning course. https://www.coursera.org/learn/machine-learning.

[15] Tensorflow. https://www.tensorflow.org/.

[16] Under the hood: Multilingual embeddings. https://code.facebook.com/posts/550719898617409/under-the-hood-multilingual-embeddings/.

[17] Wmt11. http://www.statmt.org/wmt11/training-monolingual.tgz.

[18] Wrike. https://www.wrike.com/.

[19] Judit Acs, Katalin Pajkossy, and András Kornai. Building basic vocabulary across 40 languages. In *Proceedings of the Sixth Workshop on Building and Using Comparable Corpora*, pages 52–58, 2013.

[20] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics, 2009.

[21] Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*, 2016.

[22] Mikel Artetxe, Gorka Labaka, and Eneko Agirre. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294, 2016.

[23] Mohit Bansal, Kevin Gimpel, and Karen Livescu. Tailoring continuous word representations for dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 809–815, 2014.

[24] Richard Beckwith, Christiane Fellbaum, Derek Gross, and George A Miller. Wordnet: A lexical database organized on psycholinguistic principles. *Lexical acquisition: Exploiting on-line resources to build a lexicon*, pages 211–232, 1991.

[25] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

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

[27] Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*, 2017.

[28] Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. Improving zero-shot learning by mitigating the hubness problem. *arXiv preprint arXiv:1412.6568*, 2014.

[29] Manaal Faruqui and Chris Dyer. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471, 2014.

[30] Christiane Fellbaum. *WordNet*. Wiley Online Library, 1998.

[31] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM, 2001.

[32] John R Firth. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*, 1957.

[33] Peter Gärdenfors. *Conceptual spaces: The geometry of thought*. MIT press, 2004.

[34] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[35] Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695, 2015.

[36] Dan Jurafsky and James H Martin. *Speech and language processing*. Pearson London:, 2017.

[37] András Kornai. The algebra of lexical semantics. In *the Mathematics of Language*, pages 174–199. Springer, 2010.

[38] András Kornai. Eliminating ditransitives. In *Formal Grammar*, pages 243–261. Springer, 2012.

[39] George Lakoff. *Women, fire, and dangerous things*. University of Chicago press, 2008.

[40] Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 270–280, 2015.

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

[42] Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013.

[43] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[44] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013.

[45] George A Miller and Walter G Charles. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28, 1991.

[46] Tahira Naseem, Benjamin Snyder, Jacob Eisenstein, and Regina Barzilay. Multilingual part-of-speech tagging: Two unsupervised approaches. *Journal of Artificial Intelligence Research*, 2009.

[47] Davide Picca, Alfio Massimiliano Gliozzo, and Simone Campora. Bridging languages by supersense entity tagging. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration*, pages 136–142. Association for Computational Linguistics, 2009.

[48] Dragomir Radev, Weiguo Fan, Hong Qi, Harris Wu, and Amardeep Grewal. Probabilistic question answering on the web. *Journal of the Association for Information Science and Technology*, 56(6):571–583, 2005.

[49] Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, pages 337–346. ACM, 2011.

[50] Gábor Recski, Eszter Iklódi, Katalin Pajkossy, and Andras Kornai. Measuring semantic similarity of words using concept networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 193–200, 2016.

[51] Eleanor Rosch and Carolyn B Mervis. Family resemblances: Studies in the internal structure of categories. *Cognitive psychology*, 7(4):573–605, 1975.

[52] Herbert Rubenstein and John B Goodenough. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633, 1965.

[53] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[54] Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*, 2017.

[55] Morris Swadesh. Lexico-statistic dating of prehistoric ethnic contacts: with special reference to north american indians and eskimos. *Proceedings of the American philosophical society*, 96(4):452–463, 1952.

[56] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1555–1565, 2014.

[57] Jörg Tiedemann. Parallel data, tools and interfaces in opus. In *LREC*, volume 2012, pages 2214–2218, 2012.

[58] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[59] Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011, 2015.

[60] Hyejin Youn, Logan Sutton, Eric Smith, Cristopher Moore, Jon F Wilkins, Ian Maddieson, William Croft, and Tanmoy Bhattacharya. On the universal structure of human lexical semantics. *Proceedings of the National Academy of Sciences*, 113(7):1766–1771, 2016.

[61] Kai Zhao, Hany Hassan, and Michael Auli. Learning translation models from monolingual continuous representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1527–1536, 2015.