Karlsruhe Institute of Technology

Data Selection For Machine Translation With Paraphrasing

Master Thesis of

Michael Koch

At the Department of Informatics
Institute for Anthropomatics (IFA)

Advisor: Prof. Dr. Alex Waibel

Duration:: February 1st 2015 – July 28th 2015

0.1 Zusammenfassung

Folgende Ergebnisse sind im Rahmen einer Masterarbeit entstanden. Das Ziel der Arbeit ist, Methode zu entwickeln, um besser geeignete Daten für maschinelle Übersetzungssysteme zu gewinnen. Einfach mehr Daten zu benutzen, resultiert durchaus nicht in bessere Übersetzungssysteme. Wenn eine gewisse Breite an Daten erreicht ist, muss eine inhaltliche Tiefe an Daten angestrebt werden, um Verbesserungen für Übersetzungssysteme zu erhalten.
Für viele Thematiken existieren jedoch nicht genügend Daten, die als solche gekennzeichnet sind. Techniken basierend auf Perplexität erlauben, syntaktisch ähnliche Daten aus einen allgemeinen Corpus zu extrahieren. Experimente habe gezeigt, dass diese Art der Datenselektion Übersetzungssysteme deutlich verbessert. Ein kleiner Datensatz wird dabei als Repräsentation der gewünschten Domäne betrachtet. Um noch mehr Daten selektieren zu können, wird in dieser Arbeit untersucht, wie Information über Synonyme ausgenutzt werden kann. Der domän-spezifische Datensatz wird durch Reformulierungen erweitert, sodass das Selektionsverfahren auf einem breiteren Datensatz beruht.
Von Online-Archiven deutscher Hochschulen werden Dokumente über akademische Arbeiten geladen und Textstücke extrahiert. Dazu wird ein Heuristik basiertes Verfahren vorgeschlagen und dargestellt. Das Verfahren erfolgt in einem Bottom-Up Ansatz. Ausgehend von Textboxen werden zusammengehörige Elemente ermittelt. Nur geometrische Informationen werden benutzt. Das Ergebnis ist eine Liste von Textstücken, die aneinander gehören. Da keine vollständig logische Struktur von Nöten ist, werden falsche Entscheidungen und Datenrauschen minimiert. Der so gewonnene Corpus dient als allgemeiner Datensatz, von welchem Daten selektiert werden sollen. Da dieser Corpus jedoch immer noch Rauschen beinhaltet, wird ein weiterer Corpus, zusammengestellt aus verschiedenen Quellen, benutzt, um den Einfluss des Rauschens zu untersuchen.
Information über Synonyme wird mit Hilfe zweier Arten ermittelt. Die erste Art nutzt Zuordnungen (Alignment) in multilingualen Datensätzen aus. Zurdnungsverfahren werden für Übersetzungsmodelle gebraucht, um Übersetzungspaare auf Wort- und Phrasenebene zu gewinnen. Mit Hilfe einer Pivot Sprache können Zuordnungen zwischen Phrasen derselben Sprache erstellt werden. Semantische Verwandschaftsbeziehungen lassen sich so innerhalb einer Sprache erlangen. Die zweite Möglichkeit erfolgt über rekursive Autoencoder, einer Architektur von neuronalen Netzen. Mit Hilfe eines Kodebuch für Wörter werden Satzteile gemäß eines zuvor ermittelten Grammatikbaum rekursiv gefaltet. Durch die Kodierung von Phrasen innerhalb eines Semantik-Vektor Raum, können Nachbarschaftsbeziehungen hergestellt werden und letztendlich ein Lexikon mit semantisch verwandten Phrasen-Paaren aufgebaut werden.
Die Synonymlexika werden mit zwei Prozeduren auf den domän-spezifischen Datensatz angewendet. Zufälliges Austauschen von Synonymen produziert Reformulierungen von Sätzen, in welchen gemäß einer Bernoulli-Verteilung Phrasen durch Synonyme ersetzt werden. Welches Synonym dabei genommen wird, wird durch eine Gleichverteilung entschieden, also unabhängig von Synonymwahrscheinlichkeiten. Die zweite Prozedur basiert auf das Dekodieren von statistischen Übersetzungssystemen. Statt von einer Sprache in eine andere Sprache zu übersetzen, wird innerhalb einer Sprache übersetzt. Dabei ersetzt das Synonymmodel das Übersetzungsmodel.
Zur Evaluierung werden die Kombinationen von Synonymgewinnung und Reformulierungstechniken auf Perplexität bezüglich eines der domän-spezifischen Daten ähnlichen Corpus getestet. Außerdem werden die Kombinationen auf Verbesserung in einem Englisch-Deutsch Übersetzungssystem geprüft. Paraphrasieren für Selektion erzielt marginale Verbesserungen auf dem Corpus mit Textstücken akademischer Arbeiten. Auf dem Corpus, zusammengestellt aus verschiedenen Quellen, wird keine Verbesserung durch Paraphrasierung für Selektion verzeichnet.

0.2 Abstract

With more and more data available, the focus of acquiring training data for statistical Machine Translation shifts towards obtaining more domain specific data. The challenge lays in identifying sentences which belong to a specific domain. A technique to extract text snippets from PDFs is illustrated. PDFs from the Web about academical topics are used to select from as well as a low-noisy collection of different sources.
An extension to the perplexity based selection by Moore and Lewis is presented. The small in-domain corpus is enlarged by paraphrases of its own sentences. To accomplish this extension, informations about synonyms is gathered from two different approaches. The first approach - suggested by Bannard and Callison-Burch - exploits alignments in multilingual corpora. Several criteria are defined to prune such lexica. The second approach employs recursive autoencoders to encode phrases in a semantic vector space. From positions in the semantic vector space, synonyms are derived. Two paraphrasing methods are used. Random replacement produces sentences of which phrases are replaced randomly according to some synonym lexicon. Decoding of Statistical Machine Translation allows to translate from one into another language. Here, it is used to translate within one language. A synonym lexicon serves as translation model.
All combinations of obtaining synonym informations and paraphrasing are tested in a statistical Machine Translation system as well as in perplexity comparison. Comparing to selection by Moore and Lewis, selection with enriched in-domain corpus doe yield small improvement on the corpus with academical PDFs and no improvement on the corpus composed from different sources.

Contents

1. Introduction

Language is the most sophisticated and powerful way of communication for human beings. Nowadays, many languages exist which shows its strength as well the need of humans to adapt for local conditions and individual ways of expressing themself. But languages can be barriers as well. Due to globalisation, people move more freely around in the world for reasons of business and travel. Even though there are few languages which are widespread and spoken by many people, it is still difficult to learn these languages. Most people only speak their native language and maybe an optional second language. But knowledge of languages are not only bounded to locality but also to social status. In parts of south America or Africa, neither Mandarin nor English will be of much help in speaking with locals.
An additional urgency to overcome language barriers comes from sharing knowledge. It is usually either done over the internet or by attending some school or university abroad. The most dominant language on the internet is English which also commonly serves as a lowest common denominator if it comes to speaking with foreigners. With the Internet growing, more and more people want to access content like Massive Open Online courses (MOOCs) but may not speak English in a proper way. The more specialised some information is, the more likely it is that this information will not be available in English. The same issue is valid for universities. Internationally oriented universities offer lectures in English. Universities which do not primarily aim for international students, still hold often their lectures in the native language. International students may struggle to follow if the language or the content is too complex. In particular, disciplines which have many traditional technical terms, can make international students understand less.

Engaging interpreters for such tasks will surely not be feasable. Industry and universities have started off to find solutions for these problems in an financially and qualitatively acceptable form. Usually, they all build upon Statistical Machine Translation (SMT), statistical methods to automatically perform translations between languages. Recent industrial products are *Skype translator*[1] and *Google translate*[2]. *Skype translator* supports translation for some of the most popular languages in almost real-time for text and video conferences. It not only involves actual language translation but also speech recognition. *Google translate* offers similar features. Besides textual translations, *Google translate* can be used for simultaneous translation of spoken language. It also allows to translate signs and written text captured by some camera.

[1]http://www.skype.com/de/translator-preview/
[2]https://translate.google.de/

In academia, few technical approaches exist to deal with the language issue. The European Union has funded the *EU Bridge*[3] project to bundle and focus academical research and industrial implementation for automatic language translation. Within this project, the project *lecture translator*[4] has been set up to bring simultaneous machine translation into the lecture halls. Similar to industrial products, it provides services to obtain translations for presentations, lectures and speeches. Thereby, students can watch the translation of what a speaker says in their preferred language. Another project targeting machine translation for academical use is named *TraMOOC*[5]. It is as well funded by the European Union in the *Horizon 2020*[6] program. The intention of the project is to provide usable automatic translation for specifically MOOCs and textual documents in general for European languages, Russian and Chinese as well.

These advancements in augmented reality and artificial intelligence come along with several improvements on different areas like computer vision, speech recognition or natural language processing. In particular, deep network architectures have brought enhancements in research by giving methods to acquire features automatically and not manually. Additionally, improvements in hardware allow to analyse and oversee more data than before. Latter is often referred to as *Big Data*. All of these developements have shifted artificial intelligence closer to actually comprehend and understand semantics, be it visual, textual or acoustical. Statistical Machine Translation has also profited from this progress. Different tasks of Natural Language Processing have been enhanced such as word sense disambiguation, semantic role labeling or bilingual semantic representations.
Contrary to Computer Vision, Statistical Machine Translation or Speech recognition are more crucially evaluated publicly because of the language's nature of reliable means of communication. The saying "More data are better data" holds only to some extend. At some point, new data does not generally deliver more information, especially if it does not cost much to obtain big text corpora from the internet. More domain-specific data, however allows to specialise models and hence can bring better performance. The difficulty is to harverst domain-specific data. Usually such coropora are not easily available but have to be extracted from more general or rather noisy corpora. Some approaches already exist to apply filtering methods on big general data.

In this thesis, semantic analysis will be incorporated into filtering methods to evaluate translation performance. The intention is to provide a procedure to build a Statistical Machine Translation system upon a small domain-specific text corpus and a big non-domain specific corpus.
For semantic analysis, two methods will be tested. The first method exploits alignments in bilingual corpora. Using one language as pivot facilitates associations between german phrases which are likely to be synonyms assuming they translate to the same phrase in the pivot language. The second method builds upon word embedding with skip-gram neural networks. According to a grammatical topology, a recursive autoencoder maps the folding of two grammatical entities into the same semantic vector space. Synonyms are deduced from nearest neighbours within the semantic vector space.
The lexicon of synonyms is used to generate new sentences which are semantically close to sentences in a small domain-specific text, but differ in their syntacs, that is new words and phrases, yet with the meaning preserved, are introduced. Two techniques are examined for paraphrase generation. Random replacement substitutes phrases with synonyms according to a Bernoulli distribution. The second technique employs decoding from Statistical Machine Translation considering a language model and the synonym lexicon.

[3]http://www.eu-bridge.eu/
[4]http://www.lecture-translator.kit.edu/
[5]http://www.tramooc.eu/
[6]http://ec.europa.eu/programmes/horizon2020/

Enriched versions of the domain-specific text are used in data selection from a big non-domain sepcific corpus. Two corpora are tested as non-domain specific corpus. The first corpus is taken from PDFs from academical reports and theses. An heuristic is illustrated, how to extract text from PDFs. Since this corpus is rather noisy, a second corpus collected from various sources is tested as well, in order to minimise the factor of noise in the experiments.
All combinations of semantic analysis and paraphrase generations are evaluated in an English-German Statistical Machine Translation system.

2. Related work

Various research has been done on semantics of natural languages. Roughly, two kinds of applications exist, the analysis of semantics and the synthesis of semantics. Whilst analysis is about extracting semantic informations from a corpus, synthesis handles the procedures to apply these semantic informations on some text to modify this text in some particular ways.

2.1 Semantic Analysis

Semantics origins from ancient Greek and refers to the research field of meanings in linguistics. Even though somehow related to syntax, is difficult to capture. It is as well difficult to represent semantics.

For syntax, two different kinds of representation are in use. The first representation is grammar. Rules describe the way words are adapted or changed according to which context, where words and phrases are positioned and how sentences are nested. It basically tells what a correct way of writing is, or how to group words and phrases when reading, regardless of the actual meaning. Grammar is used in school to teach languages and their structure. Computer linguists themself construct manually rules and theories for grammar representation with the intention to apply these models automatically on text. The advantages are exploiting hugely a-Priori knowledge, a minimum of internal logical errors and very detailed and precise models. It nevertheless comes with some disadvantages. Creating such models takes not only many experts but also costs time due to doing it manually. Without having the knowledge of an expert, the models cannot be created. If the input is expected to contain noise, it will perform badly because such noise would need to be modeled manually too. And modeling noise by hand will explode complexity. An alternative approach is to employ statistics. It can be performed automatically and does require very little a-Priori knowledge. Noise tolerance can be easily obtained by adapting the statistical model. Yet models will hold errors by default.
The second representation goes with empiriscm that is statistical models are built over text samples to gather information about the language or a specific domain in this language. Contrary to information which is encoded in a continuous vector space like images or sounds, the nature of text is an unbounded discrete vector space and therefore cannot be easily normalised. To make model building feasable, the *bag-of-word* concept, also known as histogram, is applied. From the text sequence, characteristic elements are extracted and their appearances is counted. *Maximum-Likelihood* is used as an estimator on how

likely a characteristic element is based on its frequency in text samples. *Markov* assumption then allows to find a probability for a specific sequence of characteristic elements. In *Computer Linguistics* n-gram models have become wildely used. Word sequences up to n words are extracted from texts as characteristic elements. To consider unseen sequences, more sophisticated methods like discounting and smoothing are emploid. N-Grams have been prooven to be simple yet powerful in many applications.

For semantic capturing, thes two approaches can be followed too. In a manually created database or ontology, these words and terms are put in relations. Such projects usually suffer from the sheer complexity of ambiguities and context as well as domain dependency. It also takes experts and time to build such databases. Some semantic relations can be found in many languages. Other terms are very specific to one language due to more abstract ways of formulating or due to the locality where the language is spoken. It is for these reasons that statistics is applied as well. Some approaches for semantic analysis exploiting statistics will be presented here.

2.1.1 Pseudo Semantic Analysis with Perplexity

Perplexity illustrates the distance of two propability distributions. In Natural Language Processing (NLP) it is used to measure how close two text are based on n-gram models[23]. Hence, it does not capture semantics but only syntax. Assuming a connection between semantics and syntax, it however can be used to find subcorpora which are syntactically close and therefore related in their semantics. The perplexity can be expressed by the cross-entropy $H(p, q)$ with an empirical n-gram distribution p given a language model q:

$$2^{-\sum_x p(x) log q(x)} = 2^{H(p,q)} \tag{2.1}$$

2.1.1.1 Selection in monolingual corpora (Moore and Lewis)

Initially, Moore and Lewis[23] presented the perplexity based semantic analysis as a way to extend some corpus, be it monolingual or bilingual. Their intention is to easily and rapidly enlarge some corpus and in the outcome better whilst constraining it to some specific domain. Usually, in-domain data is rare and obtainable only at high expense. Big general data, however, exists more frequently. But the more the training data is tailored towards the applied domain, the better the quality of the model for this particular domain will be. In a feature based decoding system, such as decoders for statistical machine translation, different language models and translation models can be included with different weights. It allows to make individual features more important if they contribute more to some domain or environment. Therefore, adaptation of the system can be reached quickly by modifying the appropriate weights. Nevertheless, this modification requires some tuning set of the targeted domain and the model of the bigger general corpus will contain more information and hence will need a larger memory footprint. Moore and Lewis suggest to extract the useful data from the large general corpus and abandon the rest. The extracted data will be added to the in-domain corpus. This way, only the small corpus will need to be trained and noise contributed by the big general corpus will be reduced.

Evaluation of information, that is phrases, shows how likely this information is in the context of the text data. Statistical n-gram models allow to model a non parametric distribution over some corpus. Hence, it can be used to compute the probability that some phrase appears in the context of the corpus which the n-gram model has been created on. The perplexity which is monotonically related to cross-entropy, delivers some measurement on how much some phrase is not expected to appear in a context of some specified corpus. As bits in the cross-entropy measurement Moore and Lewis use words, so that the cross-entropy, basically, counts the expected information for each word appearance in the context

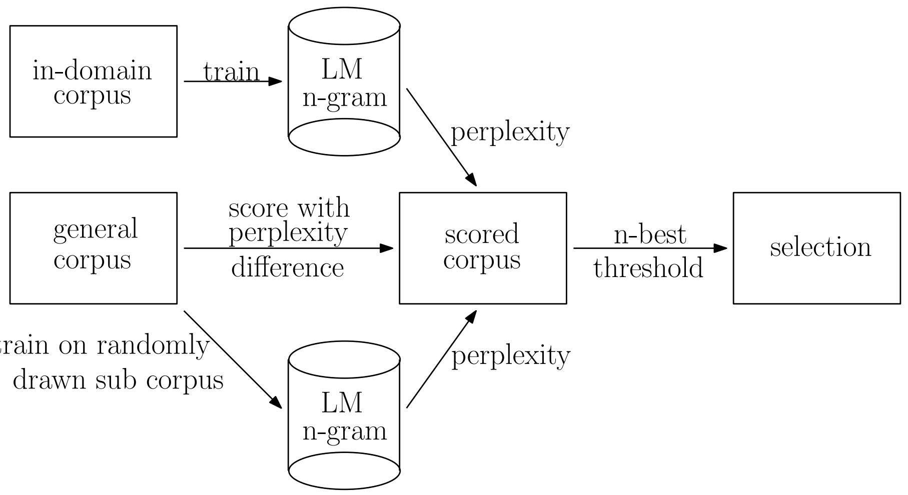

Figure 2.1: Perplexity based selection according to Moore and Lewis

of the corpus. Two models are used to evaluate the phrases. The first model is trained on the in-domain data and the second one is trained on some corpus which ideally represents non domain specific data. Having both models, a phrase can be valued how close it is to the targeted domain as well as how close it generally is to the language itself. The resulting score then is the difference between the cross-entropy on the in-domain corpus H_I and the cross-entropy on the general corpus H_G.

$$H_I(s) - H_G(s) = H(s, I) - H(s, G) \tag{2.2}$$

For the in-domain set, Moore and Lewis use parts of the English side of the English-French parallel text from realease v5 of the Europarl corpus. The general corpus is taken from the LDC English Gigaword Third Edition (LDC Catalog No.: LDC2007T07) drawing random samples such that this corpus has about the same size as the Europarl training corpus. Both corpora are preprocessed and modified such that any vocabulary, which appears at most once in the Europarl training corpus is handled as unknown token. The resulting language models are a four-gram model with backoff absolute discounting.

Moore and Lewis compare their approach with three different baselines. First one is language models trained on randomly drawn subsets from the Gigaword corpus. These subsets have similar size like the ones used for training the language models for cross-entropy prediction. Second one is using solely the cross-entropy on the in-domain corpus as score. This approach is equivalent to the method used by Lin et al[19] and Gao et al[13]. The last method is proposed by Klakow[16]. Each sentence in the Gigaword is scored by the difference in the logarithmic likelihood of the Europarl corpus according to the unigram model trained on the Gigaword corpus with and without that sentence. Varying the size of the respective training corpora, the cross-entropy on some held-out set is computed. The cross-entropy for the random selection decreases with increasing training size. For the taken training sizes, the cross-entropy is always the highest of all methods. The in-domain cross-entropy scoring decreases as well with increasing training size, but it reaches an optimal cross-entropy of 124 with a training size of 36% of the Gigaword corpus from whereon it starts to increase. The method of Klakow even outperforms this result with a minimal cross-entropy of 111 with a training size of 21% of the Gigaword corpus. However, their own approach yields better results with an optimum cross-entropy of 101 which is performed with a model built of less than 7% of the Gigaword corpus. Nevertheless, this comparison does not include the out-of-vocabulary (OOV) rate. Since different vocabulary are used even amongst the same method, the set of unknown vocabulary varies too.

Considering all models, the range of OOV rate goes from 0.75% (smallest training set based on in-domain cross-entropy scoring) to 0.03% (full Gigaword corpus). Considering only instances with minimal cross-entropy per method, the range of OOV rate is smaller, starting at 0.10% (cross-entropy difference) going down to 0.03% (random selection).

Even though, Moore and Lewis only tested on cross-entropy rather than BLEU (Bilingual Evalution Understudy) improvement with a state-to-the-art Statistical Machine Translation (SMT) decoder, there is strong evidence which supports the idea that lower cross-entropy of general corpus on in-domain corpus eventually leads to improvement in the output. Nonetheless, Moore and Lewis show that selection targeting some domain can be achieved easily and with small outcome in the end.

2.1.1.2 Selection in bilingual corpora (Axelrod et al)

Axelrod et al[1] extend the work of Moore and Lewis by applying the cross-entropy measurement onto bilingual corpora. Equivalent to Moore and Lewis, Axelroad et al argue that general parallel corpora exist much more frequently than domain specific parallel corpora. As seen before, adaptation can be achieved roughly in two ways. The first one is to operate on corpus level, that is select, join or weight corpora or subelements of corpora according to some model. The second option is to operate on model level, that is combining different models in some way. This combination is often linear where weights define the impact of a single model on the result. Contrary to Moore and Lewis, Axelrod et al try both categories. With the assumption that some general corpus contains information about the targeted domain, Axelrod et al design a technique to extract this information and to incorporate it into a state-of-the-art SMT decoder. Since in-domain data is small and therefore lack information, operating on model level may help to improve the quality. Rather than concatenating different phrase tables together, Axelrod et al follow two methods. The first method is proposed by Foster and Kuhn[12]in which they interpolate entries of different phrase tables with either linear or log-linear weights before overlapping entries are combined.

$$P(\text{target}|\text{source}) = \lambda P_{\text{general}}(\text{target}|\text{source}) + (1-\lambda)P_{\text{in-domain}}(\text{target}|\text{source}) \tag{2.3}$$

Koehn and Schroeder[17] suggest a second method. The general and in-domain phrase tables are passed on to the decoder separately, rather than combining them before decoding. The decoder keeps track of multiple decoding paths to obtain the final translation.

Moore and Lewis already presented the procedure of exploiting cross-entropy on some language model to find syntactically close data within another text corpus. This can be achieved either by by scoring according to the language model of the targeted corpus or by scoring according to the difference between the language models of the targeted corpus and some corpus which represents some general data of the language. Axelrod et al extend the cross-entropy difference approach to consider both sides of some parallel corpus. That is some arbitrary sentence or phrase is scored with the sum of the cross-entropy differences on the source and the target side.

$$(H_{\text{I-source}}(s) - H_{\text{G-source}}(s)) + (H_{\text{I-target}}(s) - H_{\text{O-target}}(s)) \tag{2.4}$$

Following Moore and Lewis, Axelrod et al ensure that the language model built on the subset of the general corpus only contains vocabulary which appears in the in-domain corpus as well. As in-domain data, they use the International Workshop on Spoken Language Translation (IWSLT) Chinese-to-English DIALOG task which is a transcription of conversational speech in a travel setting. It contains approximately $30,000$ sentencens in Chinese and English. The general corpus consists of 12 million parallel sentences put together from web data, private translation texts and publicly available datasets. For both corpora, the

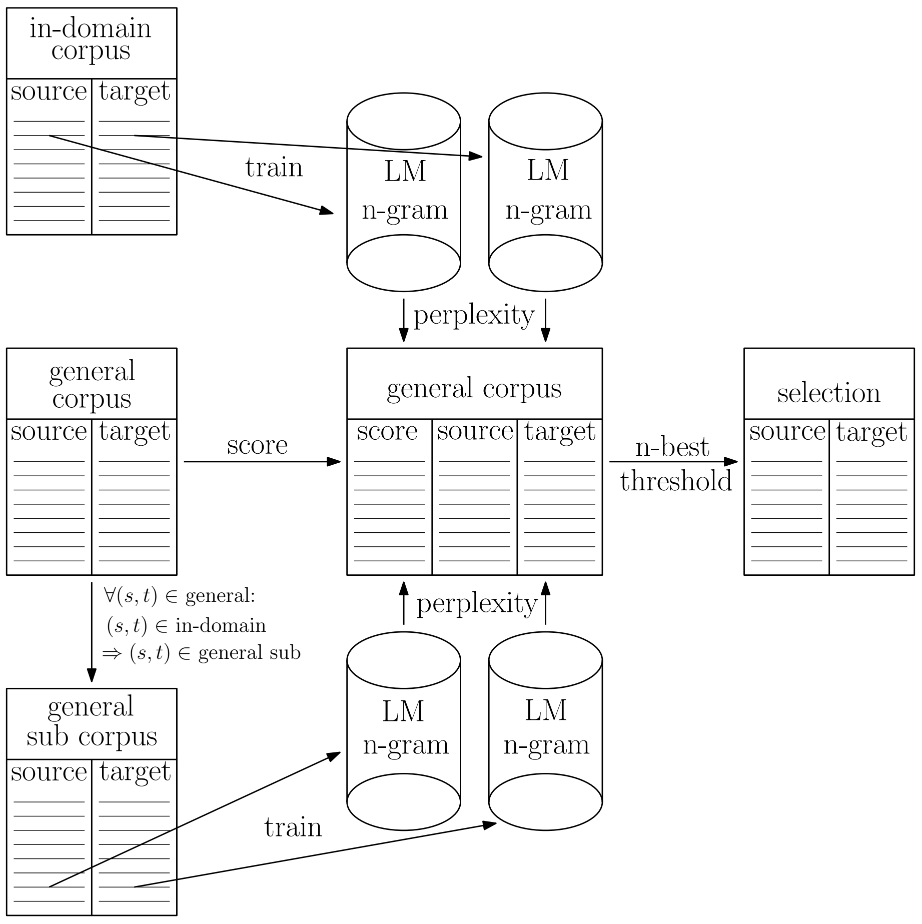

Figure 2.2: Perplexity based selection according to Axelrod et al

Method	Sentences	Dev	Test
General	12m	42.62	40.51
IWSLT	30k	45.43	37.11
Cross-Entropy	35k	39.77	40.66
Cross-Entropy	70k	40.61	42.19
Cross-Entropy	150k	42.73	41.65
Moore-Lewis	35k	36.86	40.08
Moore-Lewis	70k	40.33	39.07
Moore-Lewis	150k	41.40	40.17
bilingual Moore-Lewis	35k	39.59	42.31
bilingual Moore-Lewis	70k	40.84	42.29
bilingual Moore-Lewis	150k	42.64	42.22

Table 2.1: Translation results using perplexity based selection in Axelrod et al

Chinese side is identically segmented as well as the English side is identically tokenised. The system in use is a standard Moses framework with GIZA++[1] and MERT (Minimum Error Rate Training). The created language models are used twice, for ranking in the perplexity computation and in the machine translation process. With SRI Language Modeling Toolkit, 4-gram language models with interpolated modified Kneser-Ney discounting and Good-Turing threshold of 1 for trigrams are created. The first baseline configuration is a SMT system on IWSLT with 37.11 BLEU and the second one is a SMT system on the entire general corpus achieving 40.51 BLEU. For all three methods of perplexity based selection, the best 35k, 70k and 150k sentences are used as a training corpus for SMT systems. All three methods show results such that the selected corpora are feasable to train a productive SMT system. Using cross-entropy with 70k and 120k outperforms the baseline system trained on the general corpus. Taking only 35k delivers slightly better tests than the general baseline, although having 0.3% of its size. The method of Moore and Lewis does not show a particular improvement over the general baseline system, but needs still much less size of training corpus. Best performance comes from the bilingual Moore-Lewis approach with an improvement of 1.8 BLEU while using at most approximately 1% of the general corpus.

Even though it appears that perplexity based selection allows to find in-domain data in a general-domain corpus, there is strong evidence which supports the opposite. Perplexity of the in-domain corpus on the tuning (dev) set gives low perplexity due to the fact that the dev set has been held out from the in-domain corpus. It would be reasonable to assume that subcorpora selected with any of the three proposed methods would give similar perplexity on the tuning set. But they do not. Whilst the language model of the in-domain corpus reaches a perplexity of 36.96, the perplexities of the three methods vary from 77 to 120. To go one step further, Axelrod et al train SMT systems with the concatenation of the in-domain corpus and the selection based on bilingual Moore-Lewis. Considering that both contain related data, one could assume that they reinforce each other. In fact, results worsen if both corpora are concatenated. Axelrod et al conclude that pseudo in-domain data is selected, that is relevant data to the targeted domain but having a different distribution than the in-domain corpus. That is reason as well, to train translation models from the in-domain corpus and the selected general subcorpus separately. Both methods of combinations, as explained before, are tested. The interpolation is tested with the weight starting at 0 with an increase of 0.1 until 1. For the method proposed by Koehen and Schroeder, the two phrase tables are given directly to the decoder which are used to track multiple decoding paths. In the decoding system the phrase tables are

[1]http://www.statmt.org/moses/giza/GIZA++.html

System	Dev	Test
IWSLT	45.43	37.17
General	42.62	40.51
Interpolate ($\lambda = 0.9$)	48.46	41.28
Multiple decoding paths	49.13	42.50

Table 2.2: Translation results using phrase table combinations in Axelrod et al

Method	Dev	Test
IWSLT	45.43	37.17
General	42.62	40.51
IWSLT + General	49.13	42.50
IWSLT + Moore-Lewis 35k	48.51	40.38
IWSLT + Moore-Lewis 70k	49.65	40.45
IWSLT + Moore-Lewis 150k	49.50	41.40
IWSLT + bil Moore-Lewis 35k	48.85	39.82
IWSLT + bil Moore-Lewis 70k	49.10	43.00
IWSLT + bil Moore-Lewis 150k	49.80	43.23

Table 2.3: Translation results using in-domain and pseudo in-domain translation models for multiple decoding paths in Axelrod et al

weighted automatically on the tuning set. The two methods gain an improvement over both baselines. Best performance with the interpolation is reached with a weight λ of 0.9. The approach of using multiple decoding paths even gives an improvement of over 2 BLEU points over the baselines and 1.3 BLEU over the best interpolation instance.

In a last step, Axelrod et al connect the data selection with multiple decoding paths. Rather than using the whole general corpus, only subcorpora selected with Moore-Lewis and bilingual Moore-Lewis serve as training set for a second translation model alongside a translation model trained on the in-domain corpus. Both the in-domain translation model and the pseudo in-domain translation model are passed on to the SMT decoder. Whilst the simple Moore-Lewis approach in combination with IWSLT does not bring any enhancement compared to the baselines, combining the bilingual Moore-Lewis approach with IWSLT does. Considering approximately 1% of the original general corpus with bilingual Moore-Lewis selection returns an increase of over 3 BLEU points compared to using the whole general corpus instead, and an increase of over 6 BLEU points compared to using the in-domain corpus alone.

2.1.1.3 Selection with vocabulary (Mediani et al)

Mediani et al expand the idea of using perplexity for in-domain selection even further[20]. Their aim is to improve a SMT system targeted towards some quite small in-domain corpus such that using solely this in-domain corpus will not be feasable. Three points are suggested to reach improvement: the selection of the non-domain specific corpus, vocabulary selection for n-gram cut-offs and language model extension.

In previous proposals, including Moore and Lewis as well as Axelrod et al, a language model is created to represent non-domain specific language. Therefore, random samples are drawn from some general corpus on which the non-domain specific language model is trained. Mediani et al argue that taking randomly a subcorpus might result in having in-domain data as well for forming the non-domain specific language model. By nature, the non-domain specific subcorpus should have the highest perplexity on the in-domain language

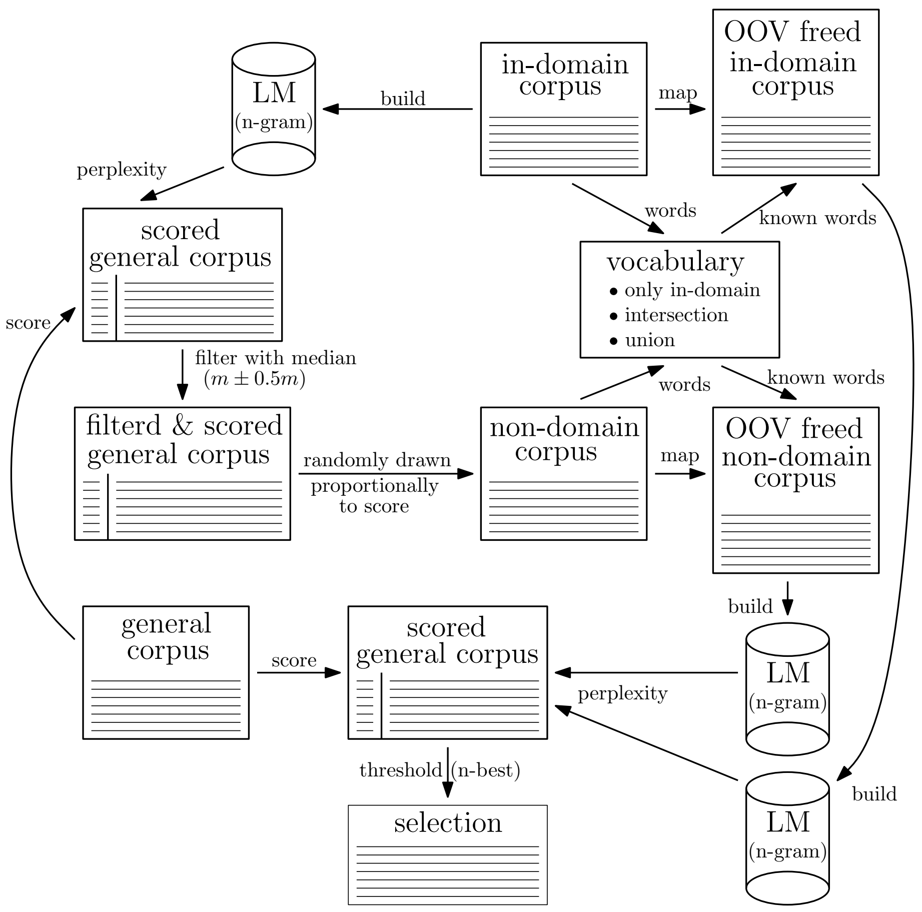

Figure 2.3: Perplexity based selection according to Mediani et al

model. Depending on the nature of the general corpus, it might include as well some useless data which will score high on any proper language model perplexity. This useless data originates mostly from web crawls or document text extraction. Mediani et al rank all entries in the general corpus to their respective perplexity on the in-domain language model. To keep useless data out of the the non-domain specific language model, some range around the median perplexity is taken. From sentences with a perplexity in range of $m \pm 0.5m$ (m being the median perplexity), Mediani et al select randomly with a probability proportional to the corresponding perplexity. For the next point, vocabulary selection for n-gram cut-offs, Mediani et al suggest additional three ways to bound the vocabulary in the in-domain and non-domain specific language models. So far, only grams which appear frequently in the in-domain corpus are considered in the language models whilst others are mapped to the unknown word tag. The first suggested bound is the intersection of grams in both corpora. The second one extends the first bound by adding the high frequency grams in the in-domain corpus. The third bound incorporates the second bound and adds high frequency grams in the general corpus. These different configurations are used not only to cut off the gram models but to also create an association lexicon between words inside the vocabulary to words outside the vocabulary. These outside-of-vocabulary words are adjoined with a certain weighting to the language model. The association between some word in the vocabulary and one outside the vocabulary is based upon a perfectly aligned parallel corpus. Assuming such perfect alignment, aligned words share some common meaning. Two words on the source side of the parallel corpus can be associated over target words which they are jointly aligned to. Using the alignment probability and Baye's rule, the joint probability $P(w_i, w_j)$ of words w_i and w_j on the source side can be expressed via the joint probability $P(w_i, z)$ of some source word w_i and some target word z. This joint probability is induced by the alignment information.

$$\begin{aligned} P(w_i, w_j) &= \sum_z P(z) P(w_i, w_j | z) \\ &\approx \sum_z P(z) P(w_i|z) P(w_j|z) \\ &= \sum_z \frac{P(w_i, z) P(w_j, z)}{P(z)} \end{aligned} \tag{2.5}$$

Since only scoring and no probability is required, it can be reduced further to the scoring function $D(w_i, D_j)$ which takes use of the alignment appearance frequency f.

$$D(w_i, w_j) = \sum_z \frac{f(w_i, z) f(w_j, z)}{f(z)} \tag{2.6}$$

Mediani et al only use unigrams in the lexicon, but this procedure is not certainly limited to this order. With the lexicon, both the in-domain and the non-domain specific language models are extended such that the out-of-vocabulary words are considered as well. Some a priori fixed probability mass m_0 is reserved by discounting the probabilities of the vocabulary words. Each word from the lexicon, then, obtains a share from m_0 proportional to the LM probability of its associated vocabulary words and the strength of the lexicon association connecting the out-of-vocabulary word with the vocabulary word.

$$P(w) = \begin{cases} m_0 P_{\text{LM}}(w) & \text{if } w \in \text{Vocabulary} \\ (1 - m_0) \sum_v t(w|v) P_{\text{LM}}(v) & \text{otherwise} \end{cases} \tag{2.7}$$

This way, a n-gram language model is obtained whose vocabulary is a superset of the original vocabulary. Only unigrams are added whilst grams of higher order remain unchanged. For the general corpus, several German monolingual corpora are choosen and reduced to

Retained Sentences in %	1	2	5	6	10
Moore-Lewis	222.7	202.4	190.3	190.0	190.5
Enhancements	211.9	195.4	185.3	184.5	185.9
Ext. Enhancements	208.1	192.9	183.4	183.3	185.0
Extension	206.2	191.9	183.0	182.5	184.4
Double Extension	203.0	189.1	181.3	181.0	183.3

Table 2.4: Perplexity on test set of LMs obtained on different techniques in Mediani et al

Retained Sentences in %	5	10	20
Moore-Lewis	13.24	13.04	12.84
Enhancements	13.47	13.19	13.06
Extension	13.52	13.16	13.00

Table 2.5: BLEU score for translation results with different configurations in Mediani et al

20 million lines (281 million tokens). For the lexicon alignment, Mediane et al use the German-English parallel corpus of public parallel corpora distributed for the WMT evaluation campaign (2014) with 3.3 million lines. From transcription of several university lectures, an in-domain corpus of 11000 lines (237000 tokens) as well a held-out corpus of similar size for perplexity evaluation is collected. Different configurations are tested. First, the extension by the lexicon is taken out which results in the configuration *Enhancements*. The second configuration uses both, the enhancement through carefully selecting the non-domain specific language model as well as extending the in-domain language model by high frequency in-domain vocabulary which is named *Ext. Enhancements*. The configuration *Extension* only extends the language models without any enhancement beforehand. Last, the language model after the enhancement step and the models after the extensions are combined, refered to as *Double Extension*. Whilst no selection gives a perplexity of 301.9, Moore-Lewis approach's results with 6% of the general corpus a perplexity of 190.0. The four configurations still excel Moore-Lewis for each tested proportion. Best perplexity is obtained by *Double Extension* with 181.0 (6%) and 181.3 (5%). Mediani et al test the translation quality with a SMT system whose translation model is trained on EPPS, NC, TED and BTEC German-English parallel corpora. Tuning and test set are parts of a computer science lecture. Tuning corpus has approximately 1000 entries whereas the test corpus incorporates about 2000 entries. First baseline is to just use the whole general corpus and the second baseline is to perform Moore-Lewis with 5%, 10% and 20% of the general corpus respectively. The first baseline reaches 12.47 BLEU. Moore-Lewis gains best with 5% of retained sentences (13.24 BLEU). Using *Enhancements* and *Extension* gives better results than both baselines. Nevertheless the improvement is slight. Best performance is reached by *Extension* on 5% of the general corpus with 13.52 BLEU.

2.1.2 Latent Semantic Analysis

Latent Semantic Analysis (LSA) is a method to map words and phrases into some vector space model (VSM) which is continuous, contrary to words and phrases themselves. The vector space represents some semantical or syntactical locality. That means that instances which are close in the vector space are close as well in their semantics or syntax. LSA works on a closed set of vocabulary. Therefore, unknown phrases and words cannot be handled. Documents are mapped linearly onto tokens. The matrix is called type-by-document matrix where type refers to the type of tokens, be it words, terms, phrases or concepts and documents be paragraphs, sentences, books, chapters or any other large text segments,

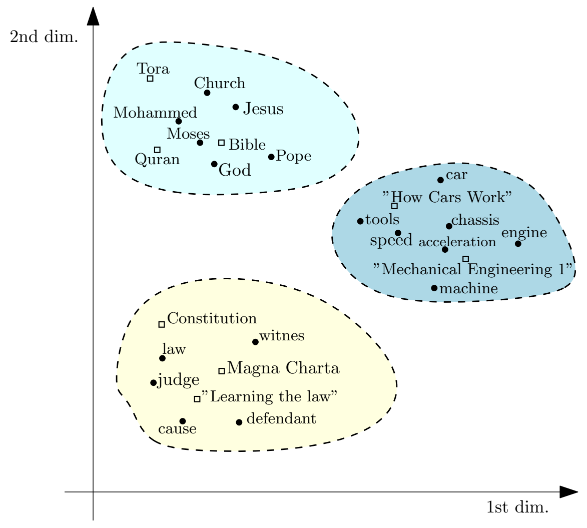

Figure 2.4: Semantic space in LSA sharing vector instances for words and books

depending on the application. The element a_{ij} in the type-by-document matrix indicates the frequency of the i-th type in the j-th document. For further improvement, a weighting function is typically applied on the elements to put more information into the system. Once, the type-by-document matrix is created, it will be decomposed orthogonaly in order to gain a mapping into some subspace which contains almost all information but still has reduced dimensionality. Single value decomposition (SVD) and Lanczos' algorithm[18] allow to perform such orthogonal decomposition. With the subspace matrix, document instances which are close to some query can be located. The query is a pseudo document, that is a sequence of words weighted to match the wanted information as much as possible. LSA is widely used to perform information retrieval within large text document systems and it shows ways to compute semantics of text.

2.1.2.1 Multidimensional Latent Semantic Analysis (Zhang et al)

An extended version of the Latent Semantic Analysis, the multidimensional latent semantic analysis (MDLSA), which combines features on a global scale with features on a local scale is proposed by Zhang et al[33]. The aim is to not only capture the statistical relationships between terms and documents but also to consider a very small context to infer a term affinity graph. Latter will help to define a spatial distribution of terms within a document. With such distribution the context of two documents should be distinguishable even if both documents have the similar term frequencies.

The global feature, Zhang et al use, is the regular LSA feature, called tf-idf weighting w_u for some token u. The term frequency f^t over all documents multiplied by the inversed document frequency idf which accounts the amount of documents where some particular term appears, results into a description following the bag-of-word concept.

$$w_u = f_u^t \cdot idf_u \tag{2.8}$$

As stated before, terms origin from a closed set, hence unknown terms - like typically in bag-of-word concepts - cannot be used and will be dropped. The closed set of vocabulary origins from a training corpus. This corpus is preprocessed, so any formated information is removed in order to obtain the raw text content. The raw text snippets are tokenised into words which are stemmed. This way, variations in grammatics are cut off to a normalised word. Any stop words like articles are removed and each resulting word u is stored with its respective term frequency f_u^t (amount of appearance u in all documents) and its document frequency f_u^d (number of documents where u appears). The inverse document frequency idf_u normalises the document frequency.

$$idf_u = log_2(\frac{n}{f_u^d}) \tag{2.9}$$

Since not all words help to deliver some improvement to the system, the basic tf-idf weighting can be used to further limitate the set of vocabulary M. Therefore, only the m-best based on tf-idf weighting are taken. Tokens with lower weights might be noise. So far, documents can be represented by some vector $x_i = [w_1, w_2, \dots, w_m]^T$ with associations w_j to the j-th word in the vocabulary M. To incorporate varieties in documents such as length and deviation from means, term weighting schemes are applied. The schemes normalise term weightings on different ways by expanding the tf-idf equation. For more details on these schemes, see parapgraph III.B of Zhang et al. The dimensionality reduction aims on finding internal structures which reflect semantic relations between documents. Though the new vector space will have considerabely lower dimensionality than the original vector space, it should be an almost accurate representation of the original one. As well, transformation between the two vector spaces should go on in a linear fashion. Each instance X in the original vector space is transformed to an instance Y in the new vector space by some linear transformation V_g.

$$Y = V_g^T X \tag{2.10}$$

The transformation matrix V_g can be computed via traditional dimensionality reduction techniques. Zhang et al use the Principal-Component-Analysis technique (PCA). PCA is designed to maximise the variance of projected vectors.

$$\max_{V_g} \sum_{i=1}^{n} \|y_i - \frac{1}{n}\sum_{i=1}^{n} y_i\|_2^2 \tag{2.11}$$

To gather spatial term distribution information, Zhang et al produce an in-depth document representation, a word affinity graph that indicates which words appear often together and which do not. The training documents are segmented into paragraphes. Since Zhang et al use HTML formated documents, they are able to exploit information from this formation to identify paragraphes in the document. Starting from the beginning, text blocks are merged subsequently until some threshold in total number of words is exceeded. If a paragraph falls below some minimum threshold, the next block is added to the paragraph. For each document i, term occurrences $g_{i,u,v}$ are counted. The word affinity graph G_i will consider the frequency $F_{u,v}$ of cooccurrence of word u and v in a paragraph, the document frequency $DF_{u,v}$ as well the term frequency f^t and document frequency f_d.

$$g_{i,u,v} = \begin{cases} F_{u,v} \cdot log_2(n/DF_{u,v})/\|G_i\|_2 & u \neq v \\ f_u^t \cdot log_2(n/f_u^d)/\|G_i\|_2 & u = v \end{cases} \tag{2.12}$$

By design, graph G_i is symmetric and stores information which will help to deduce information about local semantics. Word affinity graphes are rather spare and have large sizes. So it will be hard in terms of computation to carry out comparisons with them. To achieve feasable size for computation, the word affinity graphes need to be compressed which will allow to work accurately in a lower dimensional space. The proposed Multidimensional Latent Semantic Analysis (MDLSA) delivers this exactly. A word affinity graph G of dimensions mxm is mapped to some lower dimensional matrix Z of dimensions dxd with an mxd linear transformation matrix V.

$$Z = V^T G V \tag{2.13}$$

For reduction, Zhang et al use 2DPCA which is a variation of the Principal Component Analysis (PCA). Whilst PCA works on vectors, 2DPCA works on matrices. The objective to maximise the variance is the same. Finding the orthogonal eigenvectors of the co-variance matrix C associated with the largest eigenvalues allows to construct the optimal transformation matrix V. Nevertheless, using only the first column of Z empirically gives a good semantic representation of the respective document. Using just a vector rather than some matrix permits easy and fast comparison between documents. Contrary to latent semantic analysis (LSA) or PCA, MDLSA captures local semantic information. For feature vector comparison, Zhang et al use the cosine distance criterion which determines the magnitude of the angle between two vectors a and b.

$$\hat{S}(a,b) = \frac{a \cdot b}{\|a\|_2 \|b\|_2} \tag{2.14}$$

To compare two documents p and q which have local features p_l and q_l and global features p_g and q_g, the features similarity on global and local scale are measured separately with the cosine distance criterion and then combined linearly.

$$S(p,q) = \mu \hat{S}(p_l, q_l) + (1-\mu)\hat{S}(p_g, q_g) \tag{2.15}$$

Zhang et al design their MDLSA system for two purposes. The first one is document retrieval whereas the second is document classification. Baselines are various methods used in text information retrieval. These are:

- Multi-Level-Matching Hybrid (MLM-Hybrid)
- Multi-Level-Matching Local (MLM-Local)
- Term-Connection-Frequency (TCF)
- Principal-Component-Analysis (PCA)
- Latent Semantic Indexing (LSI)
- Vector-Space-Model (VSM)
- Rate-Adapting Poisson Model (RAP)
- Probabilistic Latent Semantic Indexing (PLSI)
- Direct Graph Matching (DGM)

Zhang et al implement two versions of their own algorithm. The first uses only local features (MDLSA) and the second comprises local as well as global features. The dataset contains text documents harvested via Yahoo Science, Web crawling and taken from other authors. A ten-fold cross-validation is performed. Metrics in use are precision, recall and AUC (Area Under the precision-recall Curve) which combines precision and recall.

$$AUC = \sum_{i_A=2}^{n_max} \frac{(Prec(i_A) + Prec(i_A - 1)) \cdot (Rec(i_A) - Rec(i_A - 1))}{2} \tag{2.16}$$

Regarding solely the AUC metric, MDLSA-Hybrid with Norm weight scheme outperforms all other methods. Noticable is that hybrid methods which means feature use on local and global scale, gain better results with respect to AUC than methods which consider only local or global features. Retrieving only few documents, MLM Local tops MDLSA methods. However, for retrieving just one document, MDLSA related methods yield best results. For document classification, Zhang et al use two datasets, YahooScience and

WebKB4. Three metrics are in use. First, accuracy is tested. It compares the amount of correctly classified documents N_C with the amount of all documents n_T.

$$\text{Accuracy} = \frac{N_C}{n_T} \tag{2.17}$$

With the F-measure F_i, both recall R_i and precision P_i of some class i are combined. The overall F-measure F_O averages F-measures of all classes by the total amount n_i of all documents in all classes.

$$F_O = \frac{\sum_i (n_i F_i)}{\sum_i n_i} \tag{2.18}$$

The last metric, Zhang et al use, is entropy E_j for some predicted class j. It measures how homogenous a predicted class is.

$$E_j = -\sum_i p_{ij} log(p_{ij}) \tag{2.19}$$

The total entropy over all classifications is a linear combination of all entropies weighted according to the relative share of documents.

$$E_O = \sum_j \left(\frac{n_j}{\sum_i n_i} E_j \right) \tag{2.20}$$

Optimising a classification task means to maximise accuracy and F-measure as well minimise the entropy of predicted classes. To perform classification, different techniques can be taken; support vector machines (SVM), Bayesian networks or neural networks can be exploited. Zhang et al take use of nearest neighbour as it is simple in use and implementation. The classification operates on the latent semantic features in the reduced space. The first dataset, YahooScience, origins from documents in the Open Directory Project. A four-fold cross-validation is performed with 25% test and 74% training data. Results are averaged over the four folds. As seen in the retrieval task, hybrid methods deliver best results. Considering traditional techniques, MDLSA-Hybrid NORM gives an accuracy boost (over 7% compared to VSM and 5% compared to LSI). Both local and global features contribute to the improvement as the linear combination parameter for interpolating global and local features in MDLSA-Hybrid NORM is approximately 0.45. Compression of data as well yields some enhancement. With over 20% accuracy gain, MDLSA outperforms DGM which operates on word affinity graphs without any dimensionality reduction. The second dataset, WebKB4, is a filtered corpus collected from university computer science departments. As before, the dataset is split in 25% test and 75% training data and results come from a four-fold cross-validation. MDLSA-Hybrid methods show best results. Lowering the impact of global features in MDLSA-Hybrid NORM and MDLSA-Hybrid BI-AC-BCA outperforms PCA and indicates that the local information generated by MDLSA delivers more information about classification than the global feature of PCA. Zhang et al also carry out tests on sentence rather paragraph level. Results are similar to working on paragraph level. MDLSA-Hybrid SMART obtains around 1% accuracy improvement. All the results presented by Zhang et al, show clear indication that semantic analysis profits from features describing local context.

2.1.3 Neural Networks

Neural networks are essential in the area of artificial intelligence and machine learning. The basic elements of neural networks are perceptrons. A perceptron y basically is a linear combination of input weights x_i plus some bias b applied to an activation function f.

$$y = f(b + \sum_i w_i \cdot x_i) \tag{2.21}$$

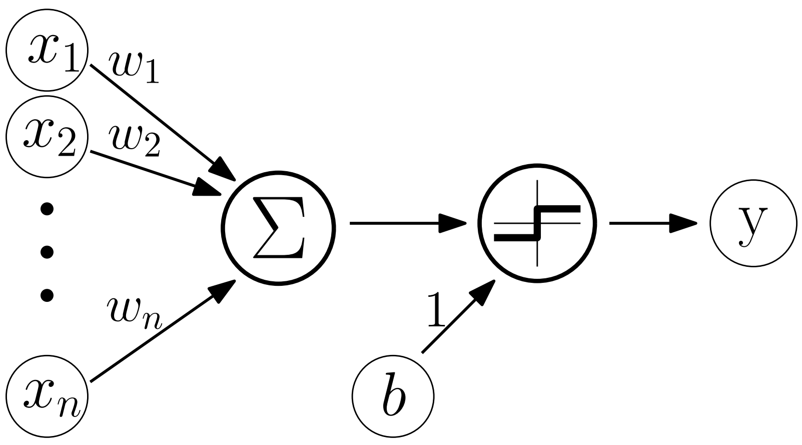

Figure 2.5: A perceptron takes an input vector $x = (x_1, x_2, \ldots, x_n)$ and feeds the weighted sum plus a bias b to an activation function f which gets activated (that is returns one) for an input greater zero and gets inactivated (that is zero or minus one) for an input less than zero.

The activation function allows to incorporate non-linearity into the perceptron. A perceptron itself is rather limited to its functionality[22]. However, using multiple perceptrons in specific arrangements allows to train for any possible function. Mostly known is the Multi-Layer Perceptron architecture (MLP), also known as Feed-Forward network which consists of layers that places perceptrons parallel. Perceptrons inside a layer are not connected with each other whilst perceptrons of neighbouring layers are fully interconnected. A MLP contains at least two layers, the input and output layer. Arbitrarily many layers, called hidden layers, can be batched between input and output layer. Several other architecture exists which are partly presented in this section.

The advantage of a neural network is its ability to learn any function possible and that it has fairly simple basics. The challenge lays in mapping discrete values, aka words, phrases or sentences, onto continuous values. This task is called word embedding and it is one of several tasks belonging to the field of Natural Language Processing (NLP). Techniques of NLP offer solutions to find semantics of languages.

2.1.3.1 Probabilistic Language Model (Bengio et al)

A very basic but critical task in NLP is to model language itself. Language consists of words which are made up from some alphabet. Whilst the alphabet usually is limited, the amount of words is not. The typical question is how likely some token is in some specific context. Therefore, joint probabilities of words are trained. Considering a closed set of vocabulary V and a fixed size of context w, the amount of free parameters is exponential to the size of context, that is $|V|^w$. Every new token in the vocabulary brings exponential many posibilities. The inherent problem is to get enough data for training. This problem is called *curse of dimensionality* and it is imminent to machine learning in general. The standard technique is n-gram. It bases on the Markov assumption stating that only a fixed size of history is relevant for prediction. Histograms for phrases up to n tokens are registered and certain methods help to smooth the empirical model. As n-grams are simple and easy to construct, they lack certain information which help to deduce the probability for unseen phrases. Due to its limited context size as well as solely exploiting appearances, semantic and grammatical variations are not considered. Bengio et al[4] propose a way to overcome these two problems to a certain extend. With a distributed representation, words are mapped into a continuous vector space. These code vectors are used to learn parameters for a joint probability function. Words which are similar in their semantic or

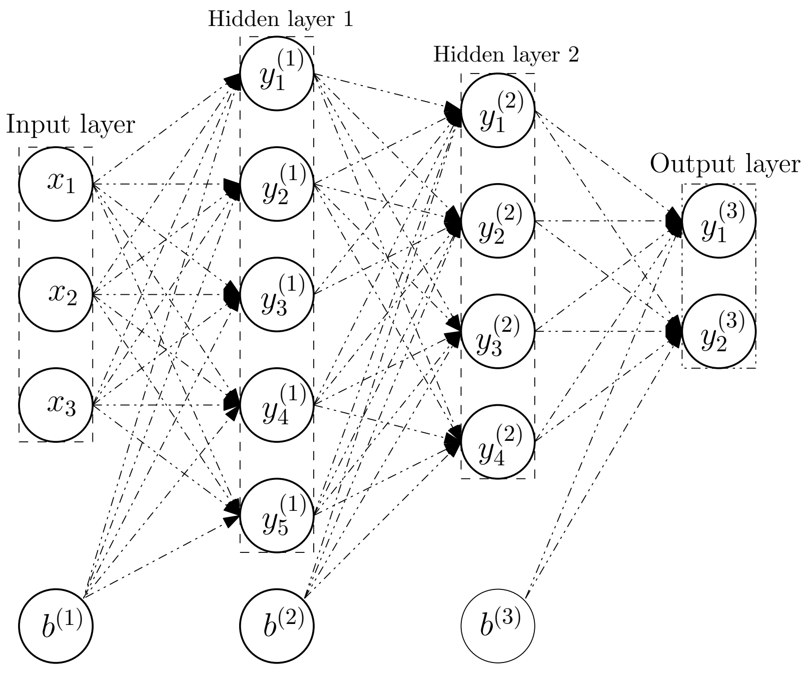

Figure 2.6: A feed forward network consists of several layers which have perceptrons in parallel and in which values (aka activations) are forwarded to the next layer. The first layer is the input, and hence is called input layer, the last one is called output layer and all layers in between are called hidden layers.

syntax, aka grammar, are supposed to be similar in the continuous vector space. Having a smoothed joint probability function, small changes in the code vectors only produce small changes in their probabilities.

The joint probability of a sequence of words w_1, …, w_T can be reformulated by the product of condition probabilities of a single word depending on its previous words.

$$P(w_1, w_2, \dots, w_T) = \prod_{t=1}^{T} P(w_t | w_1, \dots, w_{t-1}) \tag{2.22}$$

Bengio et al design a model whose objective is to learn the prediction of a word depending on previous words. Thereby, the model should only consider some limited context of size n.

$$f(w_t, \dots, w_{t-n+1}) = P(w_t | w_1, w_2, \dots, w_{t-1}) \tag{2.23}$$

Multiplying the objective function with increasing context results in the joint probability for the particular text phrase. The function is constraint to be positive and to sum up to one for all words in the vocabulary V.

$$\sum_{v \in V} f(v, w_{t-1}, \dots, w_{t-n+1}) = 1 \tag{2.24}$$

To receive the output of the objective function f, two steps have to be taken. Words must be mapped into a continuous vector space C. The mapping is provided by some matrix with dimensionality $|V| \times m$, functioning as look-up table. Second, a function g computes the condition probability distribution with word codes as input for the next word. The output of g is a vector with the i-th element estimating the probability of the i-th word being the next word.

$$f(i, w_{t-1}, \dots, w_{t-n+1}) = g(i, C(w_{t-1}), \dots, C(w_{t-n+1})) \tag{2.25}$$

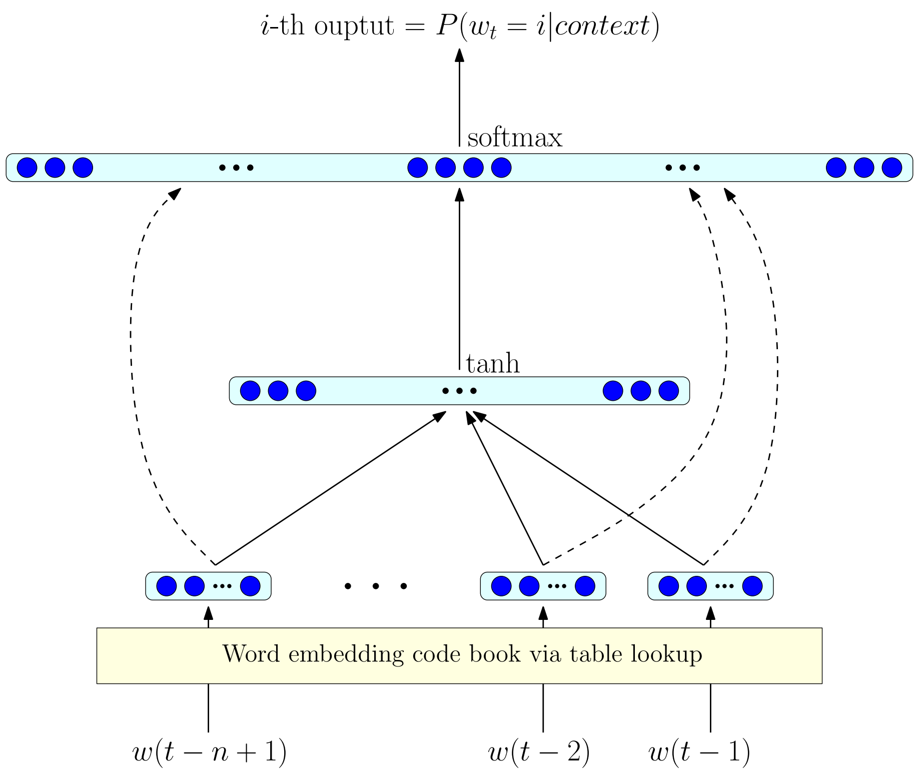
i-th ouptut $= P(w_t = i|context)$
softmax
tanh
Word embedding code book via table lookup
$w(t-n+1)$
$w(t-2)$
$w(t-1)$

Figure 2.7: The architecture proposed by Bengio et al predicts the next word based on a history of previous words.

The first mapping, the word embedding, is done by using sparse vectors, that is a vector of size $|V|$ where the i-th word is refered by the i-th unit vector. Hence, the i-th column represents the i-th word. This lookup mechanism can be seen as a single perceptron layer with no bias and no full interconnection between input and the layer, as well as the layer and output. It should be also noted for training that the lookup matrix, as free parameter, is shared amongst all input slots. The propability mapping g consists of a feed-forward network with softmax output layer. The suggested model scales linearly with $|V|$, the size of vocabulary V, and n, the size of context. With advanced models like time-delay neural nets or recurrent neural nets - Bengio et al mention - the space complexity could be pushed down to sub-linear complexity. All in all, two hidden layers, the shared word features layer C and an ordinary tangens hyperbolicus layer are used to obtain an output y which is topped with a softmax layer that guaranties positive probabilities summing up to one.

$$P(w_t|w_{t-1},\ldots,w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}} \tag{2.26}$$

Therefore, entries in the output vector y are estimations of the unnormalised log-probabilities for each word in the vocabulary. The output layer is not only fed from the tangens hyperbolicus layer but optionally from the word vector layer as well.

$$y = b + Wx + U\tanh(d + Hx) \tag{2.27}$$

The connection between the word feature layer and the softmax layer can be disabled with W set to zero. The input x of the tangens hyperbolicus layer is the concatenation of the word code vectors.

$$x = (C_1(w_{t-1}),\ldots,C_d(w_{t-1}),\ldots,C_1(w_{t-n+1}),C_d(w_{t-n+1})) \tag{2.28}$$

The training objective L is to maximise the log-likelihood of the objective function f with some regularisation term $R(\Theta)$.

$$L = \frac{1}{T} \sum_t \log f(w_t, w_{t-1}, \ldots, w_{t-n+1}; \Theta) + R(\Theta) \tag{2.29}$$

Free parameters Θ are the biases b, d and connection weights W, U, H and C. Regularisation, weight decay penalty, is performed only in the connection weights W, H and U. Performing no regularisation on the word features connection weights C might lead to unstable behaviour. Bengio et al report that however using stochastic gradient ascent such behaviour does not occur.

Bengio et al use two corpora, Brown corpus containing a text size of about 1.2 million words from English texts and books and Association Press (AP) News from 1995 to 1996 containing a text size of about 16 million words. The data is split into training, validation and test set. In particular for the Brown corpus, the training set has 800 000, the validation 200 000 and the test set 181041 words. Removing words with appearances lower than 4, the size of vocabulary is 16 383. For the AP corpus, the training set has about 14 million words, the validation set 1 million and the test set about 1 million words as well. Due to preprocessing and keeping only frequent words, the size of vocabulary is 17 964. The choosen metric is perplexity, computed with the geometric average of the reciprocal conditional probability $P^{-1}(w_t|w_1, \ldots, w_{t-1})$. For comparison, back-off n-gram models with Modified Kneser-Ney as well as class-based-n-gram models are put in place with different configurations. The neural network, designed by Bengio et al, also runs with different configurations. All neural net configurations perform better than the n-gram models. The perplexity difference between the best n-gram configuration and the best neural net configurations yields about 24% on the Brown corpus and 8% on the AP corpus. Hidden layers give an improvement which shows that latent information is extracted and helps to find the right prediction. Further more, Bengio et al also test model mixture with the neural net and a tri-gram model. Different configurations show that mixing models always helps to reduce the perplexity. A gained improvement with simply averaging the results of the neural net and the tri-gram illustrate that both models produce errors at different places. Whether the direct connection between the word feature layer and the softmax output layer has some influence, cannot be concluded from the results.

2.1.3.2 Deep Multitask Learning (Collobert and Weston)

Collobert and Weston[9] approach the field of Natural Language Processing (NLP) from a more distant point of view. The wholistic idea of fully understanding semantics of natural languages is yet to reach. Usually, NLP is divided into tasks which are handled separately, such as part-of-speech tagging, chunking, parsing, word-sense disambiguation or semantic-role labeling. Putting systems for all these tasks together will not help, as Collobert and Weston mention, because they often use simple, aka *shallow*, complexity (mostly linear ones), rely on many hand-engineered features specificly designed to the task; due to its low complexity and cascade features learnt separately from other tasks, propagation of errors is induced. Collobert and Weston suggest a general convolutional network architecture that is *deep* in terms of its complexity and only learns features which are important for the aimed tasks with very little prior knowledge. To bundle different tasks, they are all integrated into one system and trained jointly. All but the language model are trained on labeled data, thus training an unsupervised task with other supervised tasks together induces a new kind of semi-supervised learning. All in all, Collobert and Weston look into six standard NLP tasks:

- **Part-Of-Speech Tagging** (POS) estimates a syntactic label for each word such as plural noun, adverb and so on

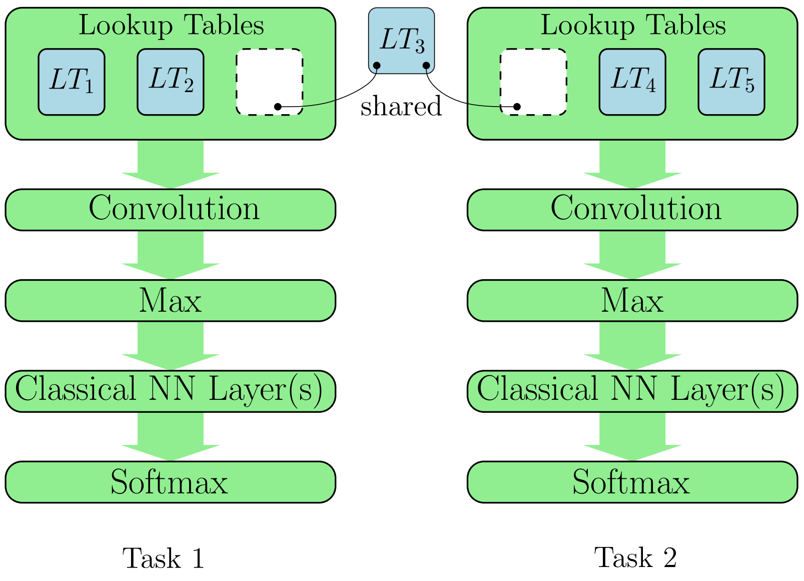

Figure 2.8: Deep multitasking works on exclusvive and shared lookup tables with own objectives for each task.

- **Chunking** parses text for grammatical roles such as noun-phrase or article
- **Named Entity Recognition** (NER) maps atomic elements in a sentence to some categories
- **Semantic Role Labeling** (SRL) estimates semantic relations of syntactic constituents of a sentence
- **Language Model** estimates the likelihood of a word sequence belonging to some language
- **Semantically Related Words** estimates the semantic relation between two words

Collobert and Weston put their attention towards SRL believing it to be the most complex task. By integrating all tasks into one system, they intent to show its general purpose as well as the improvement gained by multitask learning.

Contrary to traditional NLP systems where mostly hand-crafted features are extracted to be fed to some linear classifier like SVM, Collobert and Weston favour a deep neural network architecture consisting of several hidden layers whose objective is to extract useful features of the previous layers. Training is done on a wholistic approach which allows to make the system find features suiting best to the task. Stacking several layers, as seen in ordinary feed-forward networks, allows to generalise feature extraction layer by layer. Working with text, four issues appear.

1. Neural networks operate with continues values whilst words or any other token in text processing are discrete values. Like in the work of Bengio et al[4], words are mapped into some lower dimensioned continuous vector space via some lookup table such that each word in the vocabulary owns an instance in this vector space. Passing a sequence of words as input, the layer will provide the concatenation of word feature vectors originating from the words in the sequence.
2. Reducing data sparsity helps to avoid curse of dimensionality, that is reducing the amount of possible combinations. Since Collobert and Weston try to bring in as few prior knowledge as possible, they limit preprocessing to lowercasing and adding a feature per word indicating whether it was upper or lower cased.

3. Classification might depend on additional information which is passed on to the neural network. In SRL, labels depend on a given predicate, stating which word should be labeled. An additional lookup tables encodes the features for some word with a relative distance to the predicate.

4. Sentences have variable length of words. However, usual neural networks take a fixed length of input. A common solution is to consider only the context of fixed size. Tasks where mostly direct context influences the result show good performance with considering only a fixed-sized context. Yet SRL relies on information of far reaching dependencies. To handle a variable size of input, different techniques like recurrent neural nets or recursive neural nets exist. Collobert and Weston take use of Time-Delay Neural Networks (TDNN)[32] which are basically convolutional networks with overlapping input windows. This way, a whole sentence can be taken into account. On top, a max layer takes the maximum over time for each convoluation, so that the most relevant features of the sentence are fed forward.

Deepness in neural networks is obtained by using one or more hidden layers so the system can model non-linearity. The output layer is a softmax layer producing probabilities for each class. The whole system is trained at once, meaning that especially lookup tables at the input layer are tailored towards the classification tasks and hence are already adapted. Backpropagation is applied with the objective to minimise the cross-entropy. Next to deepness, Collobert and Weston implement multitasking in their neural network architecture. They reason that related tasks can help each other by sharing deep layers and hence improving generalisation. To do so, lookup tables are shared amongst task specific systems whilst deeper layers are separated. For training, task by task is taken for which random samples are drawn with which shared layers as well as the task-specific layers are updated. Since training samples depend on the task, different datasets can be taken if data is labeled. Besides labeled data, Collobert and Weston also exploit unlabeled data. Contrary to labeled data, unlabeled data can be gathered at considerably lower cost. A language model that predicts whether some word within some context fits in or not, is joined to the system and is able to use unlabeled data. To train the positive and negative class, a monolingual corpus is used without any modification (positive) and with replacement of random words (negative). No softmax is put on the output layer,thus regression is performed with a ranking-type cost objective which maximises the output between a positive sample and a negative one. Samples are drawn from a set S of fixed-sized sentence windows. A positive sample s is a sentence window without any modification and a negative sample s^w is a sentence window where the word in the center is replaced by some other word w.

$$\sum_{s \in S} \sum_{w \in V} \max(0, 1 - f(s) + f(s^w)) \tag{2.30}$$

For SRL tasks, a subcorpus of PropBank dataset is taken for training and testing. POS and chunking use PennTreeBank dataset, whilst NER labeled data comes from parsing the subcorpus of the PennTreeBank. The language model data is taken from the English wikipedia which then is preprocessed to eventually hold only the 30 000 most frequent words. Instances of the neural network architecture are varied to have different convolution dimensionality, amount of hidden layers and units. First, only the language model is trained and compared with WordNet, a database containing semantic relations. The coverage of their own language model is bigger than the WordNet database. Testing the SRL task, the choosen metric is the word error rate. Best performance is reached with semi-supervised learning. Testing POS and Chunking, Collobert and Weston see only modest enhancement. Nevertheless, their system does not use POS tags as input contrary to the other systems in their benchmark.

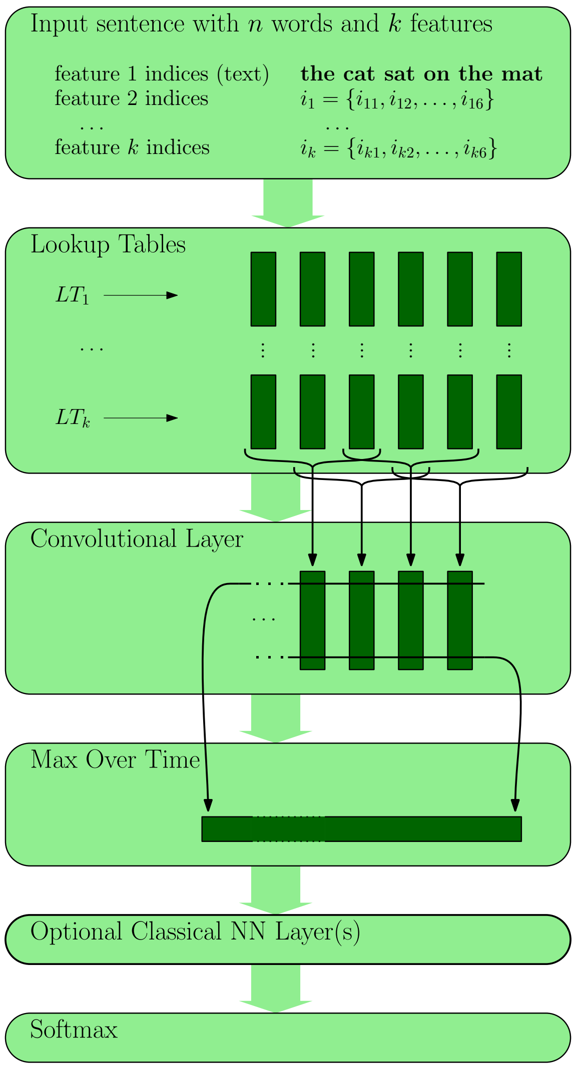
Input sentence with n words and k features
feature 1 indices (text)
the cat sat on the mat
feature 2 indices
$i_1 = \{i_{11}, i_{12}, \ldots, i_{16}\}$
...
feature k indices
$i_k = \{i_{k1}, i_{k2}, \ldots, i_{k6}\}$
Lookup Tables
LT_1
LT_k
Convolutional Layer
Max Over Time
Optional Classical NN Layer(s)
Softmax

Figure 2.9: The general concept of deep NN architecture takes an input sentence with k features like POS tags. The respective representations are taken from lookup tables for each input word and feature. In a convolutional layer, neighbouring word instances are merged together in an overlapping fasion. For each feature, the maximal value of all merges is considered as input in an optional classic neural network with eventual softmax layer on top for classification.

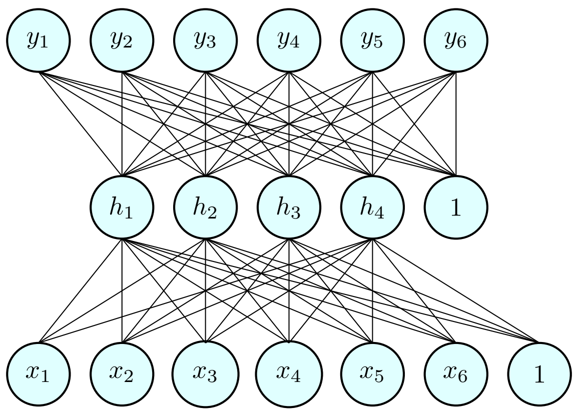

Figure 2.10: An autoencoder maps the input onto a lower dimensioned hidden layer from which it tries to reconstruct to original input layer.

2.1.3.3 Deep Learning in NLP (Du and Shanker)

A short overview about deep learning techniques in NLP is given by Du and Shanker[11]. Deep learning techniques enable feature learning over different levels of abstraction with help of multiple layers of nonlinear operations. Until 2006 deep learning was a too expensive task to perform that meanwhile is affordable thanks to new learning algorithms and hardware dedicated to massive parallel computing.

Strongly but not necessarily connected to deep learning is so-called pretraining. It coins the approach to conveniently initialise free parameters through layerwise unsupervised training. Each layer is trained with features put through the previous layers. Eventually, the extracted features can be used for standard supervised predictors or as initialisation for deep supervised neural networks. Two kind of pretraining are presented.

The first is a stacked-autoencoder. It is based on autoencoders that are architectures to aim for dimensionality reduction or compressed representation. An autoencoder is a feed-forward network and typically has hidden layers with fewer units than the input layer and an output layer that tries to reconstruct the input layer. Forcing reconstruction as the training objective provides a smaller representation in the hidden layers. In a stacked autoencoder training, each layer takes the extracted features from the previous layers as input and tries to reconstruct them via a hidden layer. The hidden layer then will be put on top for training the next layer.

The second method of pretraining is a deep boltzmann machine. It is based upon restricted Boltzmann machines (RBMs), a generative two layer architecture where the second (hidden) layer tries to reconstruct the first (input) layer. Layer for layer, the feature activations of the previous layers are taken as input to train the next layer in the stack. Eventually, all Boltzman machines are unrolled to get a deep feed-forward network which then is fine-tuned using standard backpropagation with reconstruction error.

Deep learning allows multitasking and transfer learning of related tasks which means that different systems with different objectives share some layers and thus transfer learned information. Sharing features not only enhances generalisation but also might deliver information that would not be extracted without multitasking. Empirical applications so far have shown very good results.

2.1.3.4 Word embedding (Mikolov et al)

Contrary to previous language models, Mikolov et al propose a new approach to modeling language based on neural networks[21]. They, as well show extensions and improvements

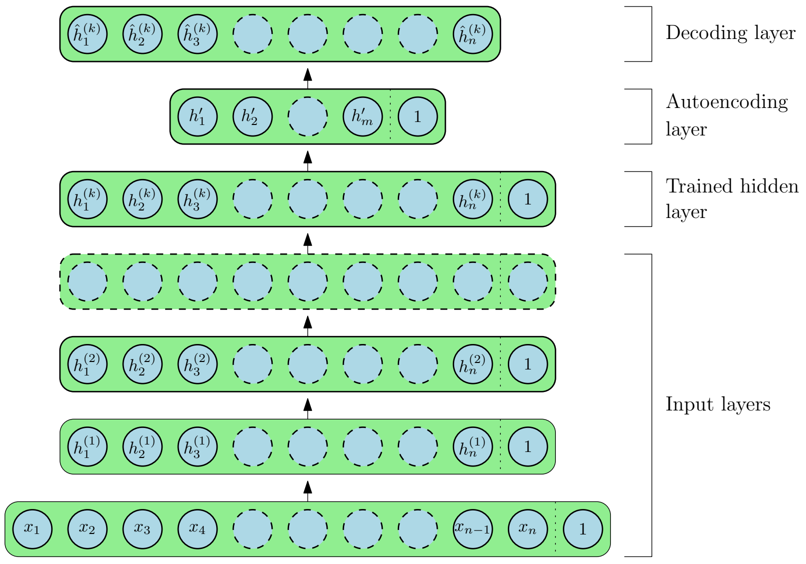
Decoding layer
Autoencoding layer
Trained hidden layer
Input layers
$\hat{h}_1^{(k)}$ $\hat{h}_2^{(k)}$ $\hat{h}_3^{(k)}$ $\hat{h}_n^{(k)}$
h'_1 h'_2 h'_m 1
$h_1^{(k)}$ $h_2^{(k)}$ $h_3^{(k)}$ $h_n^{(k)}$ 1
$h_1^{(2)}$ $h_2^{(2)}$ $h_3^{(2)}$ $h_n^{(2)}$ 1
$h_1^{(1)}$ $h_1^{(1)}$ $h_3^{(1)}$ $h_n^{(1)}$ 1
x_1 x_2 x_3 x_4 x_{n-1} x_n 1

Figure 2.11: A stacked autoencoder trains the hidden layer sequentially with an autoencoder which tries to reconstruct the last layer in the stack considering the previous layers as input layer.

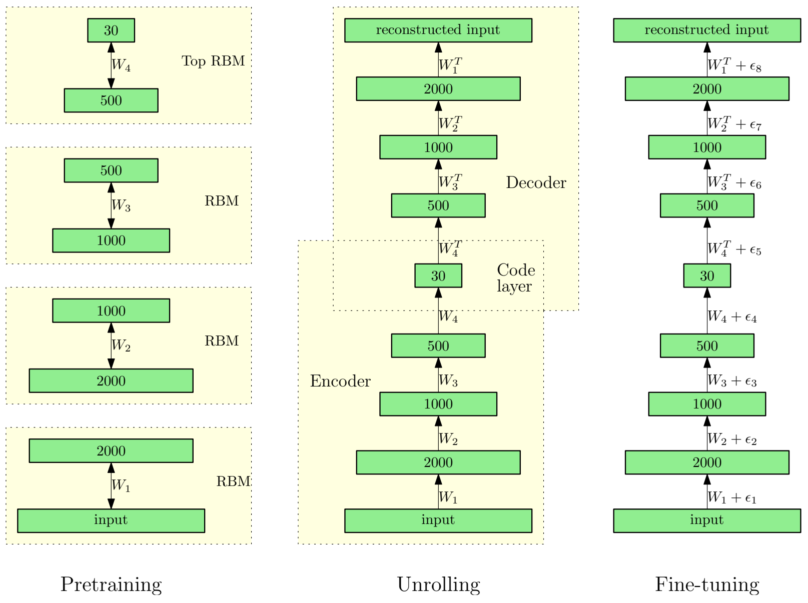

Figure 2.12: In a stacked RBM, different layers are pretrained sequentially from the bottom (input layer) up to the top layer. Pretraining gives initialisation for weights. After pretraining, all layers are stacked and with backpropagation, the weights get tuned through the whole layer stack.

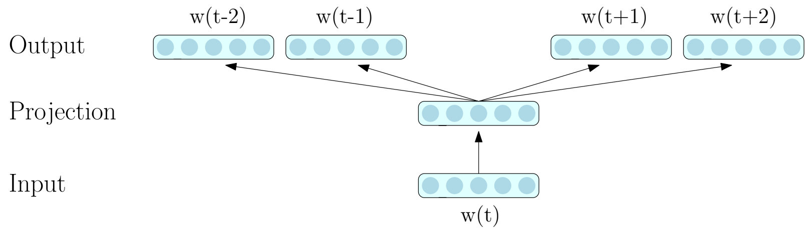

Figure 2.13: A skip-gram model predicts the surrounding word representations based on the input word representation.

for training. Thereby, they use Skip-gram models from their previous works. Usually, language models are trained to predict words based on previous words or the context. As mentioned before, neural nets operate with continuous values and hence words are mapped into a continuous vector space. The objective of the Skip-gram architecture is to find a representation that is most useful to predict the surrounding words of a sequence of words $w_1, w_2, \ldots, w_T$.

$$\max \frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j}|w_t) \tag{2.31}$$

For classification, a log-linear classification layer is in use. In a naive implementation, its complexity goes linearly with the size of the vocabulary W. Performing backpropagation will be not feasable due to the typically big vocabulary set W.

$$p(w_j|w_i) = \frac{\exp(v(w_j)^T v(w_i))}{\sum\limits_{w_k \in W} \exp(v(w_k)^T v(w_i))} \tag{2.32}$$

Computationally more efficient is an hierarchical softmax layer, first introduced by Morin and Bengio[24]. It is an approximation of the full softmax-layer, aka log-linear classifier. A binary tree represents the layer with the vocabulary words as its leafs and branch nodes containing the relative probabilities of its child nodes. Defining random walks allows to assign probabilities to words. Due to its hierarchy just logarithmically many of the words in the vocabulary have to be evaluated which gives it better time complexity than the standard formulation of the softmax layer. As reported, the structure of the tree used in the hierarchical softmax layer effects performance and the accuracy of the resulting model. Mikolov et al use a binary Huffman tree. Length of codes are distributed according to the entropy of the words, that is frequent words are assigned to short codes whilst rare words are assigned to longer codes. This will indeed fasten the training. An additional speed up can be gained through grouping words by their frequency.

Another approach to efficiently replace the standard softmax layer is Noise Contrastive Estimation (NCE). The intention is to train a model such that it distinguishes actual data from noise. NCE is able to maximise the log probability of softmax. Nevertheless, Skip-gram model is designed to produce word representations of good quality. Any simplification on NCE is allowed as long as word representations retain their quality. The simplified objective function to maximise is named Negative Sampling (NEG) and rewards correct linear classification, smoothed with sigmoid function σ for logistic regression and punishes positive classification of random words drawn from some noisy distribution $P_n(w)$. NEG only considers samples and does not require to approximately maximise the log probability of softmax. However, NCE works both on samples and numerical probabilities of the noise distribution which shows the enhancement of NEG.

$$\log\sigma({v'_{w_O}}^T v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log\sigma(-{v'_{w_i}}^T v_{w_I})\right] \tag{2.33}$$

Dependig on the size of the training set, the amount k of negative samples is choosen. Empirically, Mikolov et al see that a range of 5 to 20 is appropriate for small training set whilst for a large training set two to five samples deliver good results. The noise distribution $P_n(w)$ is a free parameter both in NEG and NCE. Mikolov et al determine the unigram distribution $U(w)$ raised to the 3/4rd power to give best results comparing to the unigram and the uniform distributions. This counts for NCE as well as NEG on every task.

A last improvement is realised by Mikolov et al regarding very frequent words. The most frequent words in a text are usually functional words, which are words that have almost no semantical information but exist for grammatical reason like "the", "or", "and", "a", "is" or "are". It also does not make a great difference to spend much training time on high frequent words after having already trained much then. For these reasons, a heuristic method is applied to subsample the vocabulary. Each word w_i in the vocabulary is discarded with the probability $P_{\text{Discard}}(w_i)$ that takes the word frequency $f(w_i)$ and some threshold t into account.

$$P_{\text{Discard}}(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \tag{2.34}$$

The threshold t is choosen to be typically around 10^{-5} and does improve accuary as well as training time. To evaluate the Hierarchical Softmax (HS), Noise Contrastive Estimation, Negative Sampling and subsampling, the analogical reasoning task, introduced by Mikolov et al [21], is performed. It contains analogies such as "Germany" to "Berlin" as "France" to unknown. The challenge lays in finding some word w (best case would be "Paris") so vec("Berlin") − vec("Germany") + vec("France") is closest to vec(w) with cosine distance (of course $w \notin \{\text{Berlin}, \text{Germany}, \text{France}\}$. Semantic analogies like the mentioned country-capital relationship are contained as well as syntactic analogies like adjective-adverb antonyms, e.g. "quick" : "quickly" :: "slow" : "slowly". The Skip-gram models are trained on a data comprising various news articles of about one billion words. After cutting off all words that appear less than five times, the resulting vocabulary has an approximate size of 692K. Negative sampling gives best results both with and without subsampling. Adding more negative samples enhances the semantic accuracy but not the syntactic. Undertaking subsampling helps to improve the semantic accuracy of negative sampling; however, it diminishes the syntactic accuracy. Subsampling significantly reduces training time to a factor of a third.

Mikolov et al address the problem of phrases. Sometimes phrases build a semantic entity that cannot be expressed by the semantics of its components. A computational issue is to figure out which combination or n-gram forms such semantic entity. In theory, all n-grams can be built and tested, but this procedure will most certainly exhaust the memory fast. Mikolov et al limit themself to bigrams and a simple approach which scores two words w_i and w_j with some discounting coefficient δ.

$$\text{score}(w_i, w_j) = \frac{\text{count}(w_i w_j) - \delta}{\text{count}(w_i) \times \text{count}(w_j)} \tag{2.35}$$

The discount δ influences how frequent words have to be grouped together. Phrases with a score above some specified threshold are merged. To allow for phrases with more than two words, several runs with decreasing threshold are carried out. With the same dataset of testing the skip-gram model, Negative Sampling and Hierarchical Softmax with Huffman coding are tested on phrase building. Without subsampling Negative Sampling exceeds Huffman. Performing subsampling, Huffman coding significantly improves and tops Negative Sampling. Accuracy can be improved further by increasing the training set. A training set of 33 billion words gives an accuracy of 77%. Lowering the size to 6 billion reduces

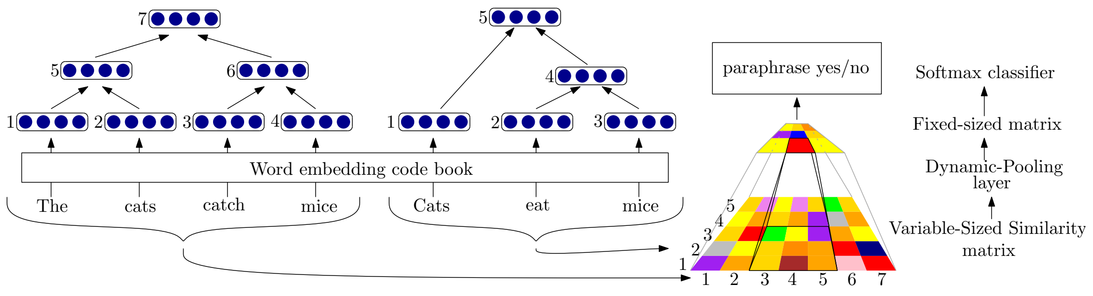

Figure 2.14: With a recursive autoencoder, semantic representations of subphrases are harvested and differences between representations of both sentences are placed in a variable-sized similarity matrix which is scaled to a fixed-sized matrix via dynamic pooling. In a last instance, a softmax classifier decides based on the fixed-sized matrix whether the two sentences are paraphrases or not.

accuracy to 66% as well.
There are interesting relations in the word vector representations. The analogy relation allows to deduce linear structures as mentioned before. The vector distribution contains another linear structure. Adding local information and some topic words results in phrases that are examples of this topic at the specified local place. When adding ”French” and ”actress”, so Mikolov et al report, the closes phrases are ”Juliette Binoche”, ”Vanessa Paradis”, ”Charlotte Gainsbourg” and ”Cecile De”. Their explanation is that the training objective is to find a distribution of context of some specific word w which is expressed by the vector instance of w. Words of the context of w are related logarithmically to the probabilities computed by the output layer. Hence, adding two words of this context is analogous to multiplying the respective context distribution. The result will be a soft AND operation of both contexts, meaning that words, that have high probabilities in both context distributions, will be scored high whereas words with low probabilities will be scored low.

2.1.3.5 Paraphrase detection based on recursive autoencoder (Socher et al)

The next step from semantics of words towards semantics of phrases is taken by Socher et al[28]. They propose an unsupervised recursive neural network or - to be more precise - a recursive autoencoder, that delivers a sequence of feature vectors used with a dynamic pooling layer for mapping on a fixed dimensioned vector space. This feature vector then is fed into a classifier to determine whether two sentences are paraphrases or not.

Paraphrase detection is about deciding whether two phrases carry the same meaning. Various tasks like information retrieval, question answering, plagiarism detection and evaluation of machine translations profit from paraphrase detection. Comparing phrases on their semantic level is rather difficult. First, phrases have arbitrary lengths. Semantics also appear at different positions to a different extend. Second, a classificator must be trained on where to draw the line between same or approximately same semantic and different semantic. Third, a phrase can contain more than one semantic information and may still be encountered to be semantically identical or close to identical. To address semantic meaning on different phrase levels, Socher et al use a recursive autoencoder. A recursive autoencoder is applied hierarchically from the bottom up. It takes an input vector $i \in \mathbb{R}^{2n}$ and maps it on some output vector $o \in \mathbb{R}^n$. With a given tree topology, the phrase or sentence is folded. Each fold results in a vector value representing the semantic of the word sequence at its leafs. The semantic of an individual word is taken from a lookup table which embedds the vocabulary into a continuous vector space. Such lookup table can be obtained from word embedding systems such as *word2vec* from Mikolov et al[21]. The topology that describes the order of folding either origins from a grammatical parser

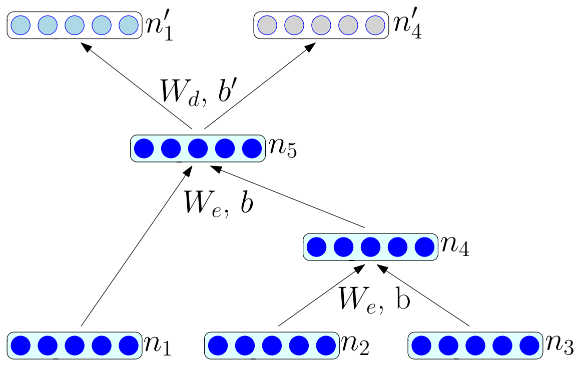

Figure 2.15: A standard recursive autoencoder encodes two nodes into one new node with shared weights W_e and bias b and computes reconstruction error on the direct decoding of child nodes with shared weights W_d and bias b'.

or is built on the fly with a greedy algorithm. In their work, Socher et al use the Stanford grammatical parser. The heuristic approach performs a fold over the nodes that gives a minimal reconstruction error. The mapping of the autoendocer is performed with the same weight matrix W_e and bias vector b for all folds for two node vectors c_1 and c_2.

$$p = f(w_e[c_1; c_2] + b) \tag{2.36}$$

As usual for autencoders, the accuracy is tested on how well the input can be decoded or reconstructed. A decoding matrix W_d and bias vector $b^{'}$ decomposes a fold vector p into two vectors $c_1^{'}$ and $c_2^{'}$. As the encoding weight matrix W_e and bias vector b are shared, so are the decoding weight matrix W_d and bias vector $b^{'}$.

$$[c_1^{'}; c_2^{'}] = f(W_d p + b^{'}) \tag{2.37}$$

The reconstruction error is the Euclidian distance between the input vectors c_1, c_2 and their reconstruction vectors $c_1^{'}$, $c_2^{'}$.

$$E_{\text{rec}}(p) = \|[c_1; c_2] - [c_1^{'}; c_2^{'}]\|^2 \tag{2.38}$$

A reconstruction error is assigned to each node in the folding tree. The training objective is to minimise the overall reconstruction error, that is the sum of reconstruction errors over all branch nodes T.

$$E_{\text{rec}}(T) = \sum_{p \in T} E_{\text{rec}}(p) \tag{2.39}$$

A length normalisation layer $p = \frac{p}{\|p\|}$ prevents weight implosion in the hidden layer. This situation appears if the autoencoder shrinks the norm of the hidden layer in order to follow the training object, stating to minimise the reconstruction error. Rather than limiting the reconstruction of a fold node to its direct child nodes, it can be totally unfolded. It then tries to reconstruct the sequence of words of its subtree. Reconstruction or decoding is carried out by recursively decoding all branch nodes in the subtree. The reconstruction error of a node p is obtained by the Euclidian distance between the word sequence w_1, w_2, $\ldots$, w_n which it spans, and its respective decoded words $w_1^{'}$, $w_2^{'}$, $\ldots$, $w_n^{'}$.

$$E_{\text{rec}}(p) = \|[w_1; w_2; \ldots; w_n] - [w_1^{'}; w_2^{'}; \ldots; w_n^{'}]\|^2 \tag{2.40}$$

To completely unfold a subtree for reconstruction error computation has two advantages. Since it tries to reconstruct each node so that it best encodes its spanned word sequence w_1, w_2, $\ldots$, w_n, it will not shrink the normal of the hidden layer. Hence, no normalisation

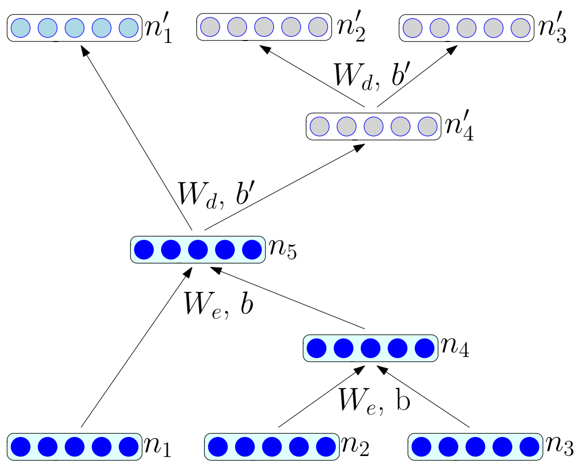

Figure 2.16: An unfolding recursive autoencoder merges two nodes into a new node with shared weights W_e and bias b. The reconstruction error is computed on the complete sub tree rather than the child nodes for which the subtree is fully unfolded with shared weights W_b and bias b'

has to be performed. Second, the more words is contained in a spanned sequence, the more important it is for reducing the reconstruction error. So, total unfold considers the size of the subtrees at its leafs and will give more regard towards bigger subtrees. Socher et al call the recursive autoencoder architecture *Unfolding Recursive Autoencoder*, whilst the previous approach is named *Standard Recursive Autoencoder*. The general architecture of recursive autoencoder can be extended to a deep recursive autoencoder. In this deep architecture, multiple levels of semantic representation, hereby called layers, exist for each node. Folding a node $x(i+1, \tau)$ in layer $i+1$ depends on both child node representations $x(i+1, \psi_l(\tau))$ and $x(i+1, \psi_r(\tau))$ in layer $i+1$ and on its node representation $x(i, \tau)$, one layer underneath. Each layer i has its own encoding weight matrix $W_e^{(i)}$ for the child nodes, an encoding weight matrix $V_e^{(i)}$ for its predecessor node and an encoding bias vector $b_e^{(i)}$ which all are shared within layer i.

$$x(i+1, \tau) = f(W_e^{(i)}[x(i+1, \psi_l(\tau)); x(i+1, \psi_r(\tau))] + V_e^{(i)} x(i, \tau) + b_e^{(i)}) \tag{2.41}$$

Uncoupling the encoding weight matrices $W_e^{(i+1)}$ and $W_e^{(i+1)}$ allows for different dimensionalities per layer. Training in general, that is deep or shallow architecture and standard or unfolding, is conducted via standard backpropagation with minimising the overall reconstruction error as training objective. The training objective is not convex. Nevertheless L-BFGS with mini-batches does a good job in converging smoothly and finding a good local optimal solution.

Applying the recursive autoencoder results in semantic representations for subphrases of the input sentence. The set of subphrases taken from the sentence depends on the topology. The extracted features need to be transformed to a new feature vector of fixed size in order to feed it to some classifier. First, a sentence similarity matrix is computed with the feature vectors of the two input sentences. It contains the Euclidian distances of the extracted feature vectors between both sentences. The similarity matrix contains word representations as well. Further on, the similarity matrix is partitioned into $p \times p$ approximately equaly sized areas. From each partition, the minimal value is taken. It is also possible to consider the average of the partition or other functions. It would be possible as well to apply overlapping regions instead of partitions. For the sake of simplicity, Socher et al only use partitioning with minimal value. Finally, the pooled matrix is normalised such to have 0 mean and variance 1.

Socher et al evaluate first the unsupervised recursive autoencoder training and then proceed to paraphrase classification. The recursive autoencoder is trained with 150k sentences from the NYT and AP sections of the Gigaword corpus. Topologies are produced with the Stanford grammatics parser. The word lookup table, aka word embedding, is taken from Turian et al[31] who use the unsupervised method of Collobert and Weston[9]. The word embedding has a dimensionality of 100. Paraphrase experiments are performed on the Microsoft Research paraphrase corpus (MRPC) provided by Dolan and Brockett[10]. For deep encoding and decoding, an additional layer of dimensionality 200 is applied. In a qualitative test, Socher et al perform a qualitative nearest neighbour analysis on standard and unfolding recursive autoencoder as well on recursivly averaging over the syntax tree-topology. Deep recursive autoencoders are left out due to their poor performance. Sentences from the Gigaword corpus are embedded into the vector spaces and representations of random phrases are taken to find the nearest neighbour in this space. Whilst recursive averaging tries to match the two last words of the current phrase in the tree. Standard recursive autoencoders (standard RAEs) performs quite well, nevertheless tends to put its focus on the last merge. Closest comes the unfolding RAE to obtain the correct syntax and semantic information. Socher et al undertake a decoding process to see how well a sentence can be decoded. Sentences encoded with an unfolding RAE are decoded. Up to five unfolds, the syntactic and semantic information is almost retained which shows the capacity of representation. In a last analysis, Socher et al examine the correctness of paraphrase detection. A 10-fold cross-validation on the training set delivers parameters for regulation and the size of the pooling matrix. The size of latter is slightly less than the average sentence length. A performance improvement of 0.2% is gained by adding the symmetric pairs to the training set, which means that for each pair (S_1, S_2) in the training set S the pair (S_2, S_1) is added as well. First, recursive averaging, standard RAEs and unfolding RAEs with dynamic pooling on top are tested. For each three of the unsupervised systems, a cross-fold validation is performed to obtain the best configuration. The less powerful systems, recursive averaging and standard RAE, yield an accuracy of 75.5% and 75.9% respectively. Unfolding RAE without hidden layers gives a 76.8% accuracy. Using hidden layers in the unfolding RAE lowers the performance by 0.2%. Dynamic pooling certainly helps to capture well syntactic and semantic information for comparing two sentences. Second, dynamic pooling is compared with other feature extraction. These are:

- histogram over all feature values with an accuracy of 73.0%; as it appears, word-of-bag methods do not capture global similarities that well
- set matching of all vectors with an 73.2% accuracy; only three features are used to describe whether the extracted features are identical, close or subset of each other; simple paraphrases can be detected, but more complex cases cannot be handled
- dynamic pooling without set matching with an accuracy of 72.6%; some essential information are not taken into the pooling matrix. Applying overlapped regions might help
- top unfolding RAE node with an accuracy of 74.2%; the euclidian distance is taken as feature; it certainly shows the strength of the unfolding RAE but gives clear indication that features from lower nodes are necessary too

The figures show the strength of unfolding RAEs which still needs the dynamic pooling in order to find the relevant comparisons between both sentences. Lastly, Socher et al compare their approach with techniques of previous works. Applying unfolding RAE with dynamic pooling exceeds all other approaches in terms of accuracy and F1-measure. And in contrast to other methods, it does not need manually designed features.

2.2 Paraphrase generation

As seen in the section about semantic analysis before, various ways exist to measure the semantic context of text. Besides bag-of-word approaches, most of these techniques are based on considering the context to a certain extend, mapping it into a continuous vector space that is reduced in specific ways to eventually represent semantic information. For creating paraphrases, different paths need to be followed. There are a two kinds of models: generative and discriminative models.

Generative models work with join probabilities of observation and label. Therefore, they can be used to generate synthetic observations, that is data points whose appearances are distributed according to the trained join probability. Generative machine learning systems are Naive Bayes, Restricted Boltzmann Machines, Mixture models, Hidden-Markov-Models and others.
Discriminative models work with conditional probabilities for observation given some label. The essential difference to generative models is that deducing the most likely observation in a discriminative model framework requires to know the label or the a-Priori probabilities of all labels beforehand. Often, discriminative models can be described and searched more efficiently. However, the set of possible labels is usually predefined and thus training will use it to find thresholds and hyper-planes as separation in the feature space. Unseen labels will not be considered. This might lead to an unstable and incorrect system. Typical discriminative systems are Support-Vector-Machines, Logistic Regression or Neural Networks.

With Bayes' rule, joint probability can be expressed with conditional probability and vice versa.

$$P(a, b) = P(a|b)P(b) \tag{2.42}$$

This relation can be exploited to accomplish generating data points with conditional probabilities and a-Priori distributions or to find the most likely data point given some label.

2.2.1 SMT Approach

Paraphrasing sentences can be seen as a translation within the same language where the typical bilingual translation information is replaced by paraphrased information. Additional information like language model, syntactical structure and so on, can be considered as well. As usual in statistical machine translation tasks, it is based on discriminative modelling since the input sentence is known.

Statistical Machine Translation (SMT) systems exploit the Noisy Channel model of Brown et al[7]. Given some input sentence S, the most likely translation $\hat{T}$ has a maximum a-Posteriori probability $P(T|S)$ over possible target sentences T. This a-Posteriori probability is proportional to the a-Priori probability $P(T)$ of the possible target sentence and the a-Posteriori probability $P(T|S)$ of the possible target sentence.

$$\hat{T} = \arg\max_{T} P(T|S) = \arg\max_{T} P(S|T)P(T) \tag{2.43}$$

This basic model can be extended to more complex models like the Log-Linear models which can incorporate several features and their importance.

2.2.1.1 Monolingual Machine Translation (Quirk et Al)

Tools to form and train a SMT system already exist. A standard SMT system needs a parallel bilingual corpus for training at least. Comparable bilingual corpus can be used with certain preprocessing steps. Quirk et al[27] conclude that applying these techniques on monolingual parallel corpora should deliver semantic translations.

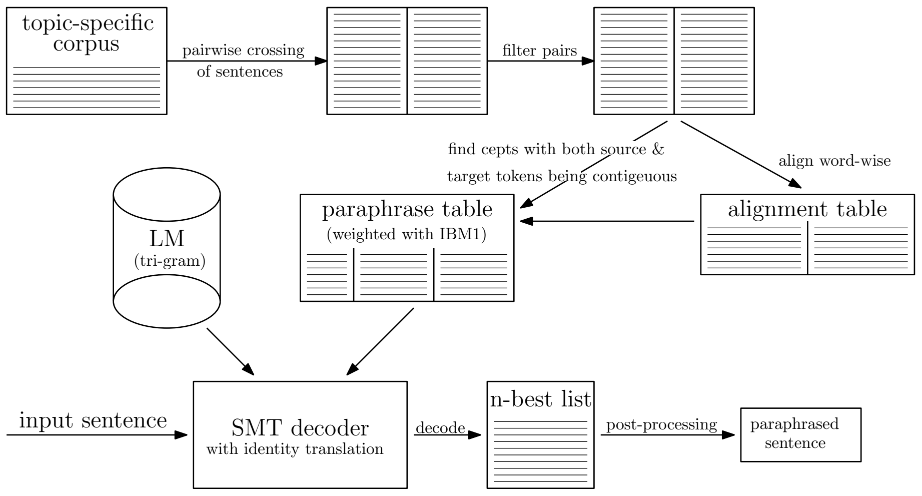
topic-specific corpus
pairwise crossing of sentences
filter pairs
find cepts with both source & target tokens being contigeuous
align word-wise
LM (tri-gram)
paraphrase table (weighted with IBM1)
alignment table
input sentence
SMT decoder with identity translation
decode
n-best list
post-processing
paraphrased sentence

Figure 2.17: Paraphrase generation based on SMT tools with monolingual corpora according to Quirk et al

Alignment	Precision	Recall	AER
Giza++	87.47	89.52	11.58
Identical word	89.36	89.50	10.57
Non-Identical word	76.99	90.22	20.88

Table 2.6: Evaluation of alignment in Quirk et al

Monolingual parallel corpora are even more expensive than bilingual parallel corpora. Comparable monolingual data, however, is quite easily to gain. From news aggregation sites like *http://news.yahoo.com* or *http://news.google.com*, Quirk et Al collect URLs of news articles which are clustered regarding different topics. Within one cluster, sentences are compared pairwise with the Levenshtein edit distance. Pairs of sentences are filtered out according to these criteria:

- sentences are identical or only different in punctuation
- duplicates
- sentences' lengths differ to much regarding the lengths quotient
- Levenshtein edit distance greater 12.0

The resulting pairs of sentences are aligned wordwise with Giza++. Alignment goes both directions which is then heuristically recombined into a single bidirectional alignment. The alignment is evaluated with a heldout cluster from which 250 sentence pairs are randomly drawn. An independent human evaluator ensures that the sentence pairs contain paraphrases. Two annotators label assignments as either *sure* (that is correct) or *possible* (that is allowed but not necessary). Conflicts are resolved by annotators firstly reconsidering their decision and secondly setting the label *sure* only if both annotators consider the case as *sure*. Otherwise the label *possible* is taken. Alignment with Giza++ on monolingual data is higher than on usual bilingual data indicating noisy training data. That is the training set is rather comparable than parallel. The AER of non-idential words falsely indicates poor support for paraphrase generation. For the paraphrase table, synonymous phrases are identified with a phrasal decoder and a probability according to IBM-Model-I is assigned. Intra-phrase reordering allows for grammatical variatons like *margin of error* replacing *error margin*. A language model is trained on the whole news corpus which includes about 24 million words. The model is a trigram model with interpolated Kneser-Ney smoothing. Generating paraphrases is implemented that a standard SMT decoding

Method	of 59	of 59 + 141
1-best PR	54 (91.5%)	177 (89.5%)
2-best PR	53 (89.8%)	168 (84.0%)
3-best PR	46 (78.0%)	164 (82.0%)
4-best PR	49 (74.6%)	163 (81.5%)
MSA	46 (78.0%)	-
5-best PR	44 (74.6%)	155 (77.5%)
WN	23 (39.0%)	25 (37.5%)
WN+LM	30 (50.9%)	53 (27.5%)
CL	14 (23.7%)	26 (13.0%)

Table 2.7: Human acceptability judgmenent in Quirk et al

Paraphrase technique	MSA (of 59)	PR#1 (of 100)
Rearrangement	28 (47%)	0 (0%)
Phrasal alternation	11 (19%)	3 (3%)
Information added	19 (32%)	6 (6%)
Information lost	43 (73%)	31 (31%)

Table 2.8: Qualitative analysis of paraphrases in Quirk et al

approach which only considers the paraphrase model. A lattice is built upon the preprocessed input sentence with all paraphrase replacements. Edges labeled by input subphrases are assigned some uniform probability u. A high probability u results in a more conservative generation, whilst a low probability u will try more paraphrase replacements. The Viterbi algorithm supports an effective way of computing the n-bests paths within the lattice. Post-processing transforms the paraphrasing sentence into its eventual form.

For evaluating the whole system, Quirk et al form corpora (sentences with corresponding paraphrases) from MSA and WordNet (59 sentences). 141 randomly selected sentences are added from held-out clusters. Paraphrases are produced with three systems: WordNet with a trigram LM, statistical cluster with a trigram LM and 5-bests phrasal replacements. which is the system from Quirk et al. The output is assessed by two human judges. On disagreed sentences, the judges reassess, which increases their agreement from 84% to 96.9%. WordNet and statistical clusters perform badly even with a language model. MSA, however, gives better results, although n-bests of phrasal replacement significantly outperform the MSA. The previously reported high AER for non-identical words is shown to be a false indicator for the paraphrasing performance. For the most input sentences, up to 200 distinct paraphrases could be produced with phrasal replacement. Quirk et al also analyse the kind of paraphrasing involved in MSA and phrasal replacement. Since phrasal replacement involves simple replacements, it might appear that MSA uses more in-depth and complex paraphrasing techniques like active-passive transformation. Quirk et al test on rearrangements, phrasal alternations and added and lost informations. At first glance, MSA shows extremly complex paraphrasing applications. However, an in-details inspection of MSA delivers template translations which can bring brilliant results if the input is semantically close to the template. If it is not the case, the output will be much more general. That is information gets lost. Simple replacements are modest in their way of paraphrasing, but consistently keep information, so far, more on hand than more complex techniques. By now, monotone decoder systems are pretty effective in producing paraphrases, as Quirk et al present. The initial data for training a paraphrase generator on monolingual data costs quite a lot and is rather sparse. Quirk et al spent about eight months on crawling and observing articles to gather their corpus. The eventual data leads to more comparable than parallel shape. For different domains than news topics, it will not be easy to find the required mas of data and range of cover. Quirk et al propose to focus more on preprocessing to filter out noise in order to to make the corpus more parallel.

2.2.1.2 Alignment via bilingual pivoting (Bannard and Callison-Burch)

Rather than monolingual parallel or comparable corpus, Bannard and Callison-Burch suggest to exploit bilingual data for paraphrasing task[2]. The second language (which is not aimed to paraphrase within) serves as pivot to find paraphrase relations between two sentence in the first linguage which are aligned to one sentence in the pivot language. As argued before, gathering monolingual parallel corpora is expensive and sparce not only in its amount but also in the coverage of domains. Bilingual corpora are neither redundant in depth or in coverage, yet there is more data than considering monolingual data. The assumption, Bannard and Callison-Burch build on,is that different phrases in one language are semantically related if they all are aligned to the same phrase in the pivot language. This is, however, converse to the idea that due to multiple meanings different phrases are aligned to the same phrase in the pivot language. Paraphrases in aligned monolingual data are concluded from matching context. Hence each alignment of different phrases delivers at most one potential paraphrase. Incorporating a pivot layer increases the amount of potential paraphrases by how many times the phrase in the pivot language has been aligned to. Alignment is performed with the heuristic algorithm of Och and Ney[25]. It incrementally detects alignment from words and phrases which have adjacent alignment points. Probabilities of alignments (s_1, s_2) between target and pivot languages are estimated with maximum likelihood.

$$p(s_1|s_2) = \frac{\text{count}(s_1, s_2)}{\sum\limits_{s} \text{count}(s, s_2)} \tag{2.44}$$

Probabilities of alignments (e_1, e_2) within the target language are derived from alignment probabilities from target-to-pivot language and vice versa and marginalising over all alignment phrases f in the pivot language.

$$\hat{e} = \arg\max_{e_2 \neq e_1} p(e_2|e_1) = \arg\max_{e_2 \neq e_1} \sum_{f} p(f|e_1)p(e_2|f) \tag{2.45}$$

Bannard and Callison-Burch extend finding the most likely paraphrase of e_1 by considering the context or sentence S where e_1 appears.

$$\hat{e} = \arg\max_{e_2 \neq e_1} p(e_2|e_1, S) \tag{2.46}$$

The context information S allows for reranking. Various models can be brought in. Bannard and Callison-Burch limit themself to a simple language model which estimates the sentence S with e_1 replaced by the potential paraphrase e_2.

The bilingual corpus for evaluation is the German-English section of the Europarl corpus, version 2. Alignment is performed automatically with Giza++ and manually as golden standard. 46 random English phrases which are in WordNet as well appear multiple times in the first 50000 sentences of the bilingual corpus. The manual alignment only corrects the selected English phrases based on the automatic alignment. Since paraphrasing depends on the context, the paraphrases are regarded inside some sentences where the original phrase appears. Two native English speakers evaluate to whether the produced sentences preserve the semantic information and is grammatically correct. Sentences which fulfill both conditions are considered correct whilst failing one condition will be labeled incorrect. The inter-annotator agreement reaches $\kappa = 0.605$. Different configurations are tested: Manual alignments, automatic alignments produced over the German-English section of Europarl

Configuration	Paraphrase prob	Paraphrase prob and LM	Correct meaning
Manual Alignments	74.9	71.7	84.7
Automatic Alignments	48.9	55.3	64.5
Using Multiple Corpora	55.0	57.4	65.4
Word Sense Controlled	57.0	61.9	70.4

Table 2.9: Paraphrase accuracy and correct meaning for different configurations in Bannard and Callison-Burch

and over multiple corpora indifferent languages. Optional reranking with language model and optional candidate paraphrases limitation to the same sens as the original phrase are applied too. The baseline reaches an accuracy of 74.9%, that is grammatically and semantically correct. Disregarding grammatics, accuracy is enhanced to 84.7%. This increasement indicates that context information is rather important for grammatics. Reranking paraphrase candidates with a trigram LM model makes the context to influence paraphrasing. The result shows a small decrease and suggest that manual alignment already considers the context. Automatic alignment performs worse, giving only 48.9% accuracy considering semantics and grammatics. Leaving out grammatics shows improvement to 64.5% like in the manual alignment. Applying the trigram LM noticeably raises accuracy to 55.3%. More information about context and grammatics might help to improve the performance of automatic alignments. To overcome the sparsity of data, multiple pivot languages are used and ranking is adopted accordingly to marginalise over a set C of parallel corpora of different language combinations.

$$\hat{e} = \arg\max_{e_2 \neq e_1} \sum_{c \in C} \sum_{f \in c} p(f|e_1)p(e_2|f) \tag{2.47}$$

Bannard and Callison-Burch exploit the French-English, Spanish-English and Italian-English section of the Europarl corpus as well. Approximatelly 4000000 sentence pairs are used for training. Paraphrasing without LM reranking significantly imrproves accuracy up to 55.0%. Using a LM for reranking as well considering only semantics do not show big improvements comparing it with using only German-English data. More data apparently results in better understanding of grammatics. To avoid ambiguities which might result in false paraphrases, candidate paraphrases are filtered in order to preserve meaning. Considering the pivot phrase as well gives a mechanism to avoid ambiguities. Word sense controllation not only hands out grammatical information but also carries more semantic information. Without LM, accuracy reaches 57.0%. Incorporating a LM increases accuracy to 61.9%. Disregarding grammatics delivers an accuracy of 70.4%.

Bannard and Callison-Burch illustrate an approach on how to exploit bilingual data for paraphrase generation. Whilst it offers some advantages over the monolingual approaches, there is still space for improvement. In particular, modeling grammatics would increase performance.

2.2.1.3 Machine Translation Within One Language as a Paraphrasing Technique (Barancikova and Tamchyna)

Exploiting paraphrasing, Barancikova and Tamchyna develop a method of enriching machine translation evaluation[3]. Their aim is to make automatic evaluation of MT systems more accurate. In particular, BLEU score serves as baseline which then will be extended to take use of paraphases. Since actual translation is not deterministic and can be seen as some form of interpretation, arbitrarily many translations are possible. Human judges are able to qualitatively determine the correctness of translations. However, automatic evaluation relies on a set of reference translations. By nature, this set will never be exhausting.

BLEU counts how many phrases overlap between the candidate and reference translations. Only direct matches are considered but no synonyms or paraphrases. If the set of reference translations contains only few entries, BLEU will perform badly. Barancikova and Tamchyna aim on enriching the set of reference translations by adding paraphrases. Since paraphrasing can be seen as a translation within one language, standard tools for SMT are adapted and applied.

Barancikova and Tamchyna work on data from the English-Czech translation task of WMT12. Czech paraphrases are taken obtained from the Czech WordNet 1.9 PDT and the Czech Meteor paraphrase tables. Whilst Czech WordNet 1.9 PDT has only few but qualitatively good paraphrases, the Meteor Paraphrase tables are large but quite noisy. Noise is reduced with folowing scheme:

1. Only pairs consisting of single words are kept; Barancikova and Tamchyna are not able to reduce noise on multi-word paraphrases.
2. Morphological analysis; word forms are replaced by their lemmas.
3. Pairs with same lemmas are removed.
4. Pairs whose words differ in their part of speech are removed.
5. Pairs with unknown words are removed.

In case of numeral and corresponding digits, the last two rules are not active. After applying these rules, the Meteor corpus is reduced from almost 700k pairs of paraphrases to just 32k pairs. Moses is used as SMT system in phrase-based setting. A language model is trained on Czech part of the Czech-English parallel corpus CzEng[6] with SRILM. Two phrase models are trained. Each model contains phrases, their translations and several feature scores like transition probability or lexical weight. These models can be produced from large parallel data. Due to the scarceness of large parallel Czech-Czech data, the required data will be synthesised. The first phrase model is created from the Czech Paraphrase Metero table. Using pivot languages according to Bannard and Callison-Burch[2] allows acquiring paraphrases based on alignments. Since the pivot score does not serve well, new paraphrase scores are introduced based on distributional semantics. The context similarity of paraphrases are measured via cosine distance. In total, six scores are used according to the context window (one to three words) and considering word order. For the second phrase model, a set of words which appear more than five times in the Czech part of the CzEng corpus is merged with the set of words appearing in the MT outputs and reference sentences. Next, a morphological analysis is run on the words in the resulting set. Every pair of words which fulfills one of the following four conditions, is added to the phrase model for each word x from the set:

- (x, x) (allows to suspend paraphrasing for x)
- (x, y), if lemma of x is lemma of y (morphological variation is a way of paraphrasing)
- (x, y), if lemma of x and lemma of y are paraphrases according to Czech WordNet PDT 1.9
- (x, y), if lemma of x and lemma of y are paraphrases according to the filtered Meteor.

The first four scores determine whether the respective condition is fulfilled. A fifth score expresses POS similarity between the two words. Besides language model and phrase models, an additional feature is used to make the MT decoder favour output closer to the hypotheses. It counts words in the hypothesis which is confirmed by the reference translation. Their implementation not only incorporates the feature into the beam search but also gives an estimator for future phrase score which is the number of reference translation

Configuration	correlation	avg BLEU
Baseline	0.75	12.8
Paraphrased	0.50	15.8
LM+0.2	0.24	9.1
LM+0.4	0.22	6.7

Table 2.10: Configurations and their Pearson's correlation of BLEU and human judgment and additionally average BLEU score in Barancikova and Tamchyna

Configuration	correlation	avg BLEU
Lexical	0.56	15.1
Lexical and LM+02	0.33	9.5
Monotone	0.61	18.1

Table 2.11: Additional Configurations and their Pearson's correlation of BLEU and human judgment and additionally average BLEU score in Barancikova and Tamchyna

words covered by the given phrase. Parameters are tuned with minimum error rate training (MERT) on the reference sentences on the highest rated MT output. This approach does not deliver good weights. The language model is weighted low whilst the feature for steering towards the hypothesis is estimated to be very important. Barancikova and Tamchyna start off setting parameter weights by hand.
Four different configurations are compared on the Pearson's correlation of BLEU and human judgment as well the average BLEU score. Using paraphrases does not yield an improvement. Even though applying paraphrases with automatically tuned weights gives a better BLEU score, the correlation to human judgment decreases. Most of its output is not grammatically correct. Increasing the weight for the language models rectifies the quality of grammar but also reduces the semantic information. Both phrase models and targeting feature introduce noise. The former might need more preprocessing and better descriptive scores. The latter is not sophisticated enough to know good translation and rather prefers to have as many word matches with the reference translation as possible. Dealing with the poor results, Barancikova and Tamchyna test three additional configurations in which the Enhanced Meteor table is omitted due to its noise. The last configuration goes without reording using monotone decoding. In particular, monotone decoding shows improvement of correlation with increasing BLEU score. However, none of these configurations reaches the baseline.

2.2.2 Randomised Aproach

SMT offers one way of transforming a sentence into a paraphrased version of itself. Therefore, a lattice is built upon the input sentence to express possible translations. Evaluation of all paths within the lattice requires the decoder to have exponential time complexity. Usually heuristics like beam search and future cost estimation are applied to make the decoding process more feasable. Since paraphrase generation is a translation into the original language, not all words or phrases need to be translated. Adding identity translations allows to suspend translations whilst applying SMT decoders which are designed to translate all but phrases for which no translations are known.

A different approach to SMT decoding is the use of randomised algorithms. They include randomness into their logic to achieve a probabilistic solution. There are two kinds of randomised algorithms.
Monte-Carlo algorithms are bounded in their time complexity but produce results which are good or correct with a certain probability. This is usually achieved by taking random

input several times and returning the averaged or best outcome.
The second kind of randomised algorithms is named Las-Vegas-algorithm. Randomness in this case refers to time or space complexity. A Las-Vegas algorithm always returns a correct answer on some random input but with reduced expected time or memory consumption.

2.2.2.1 Monte-Carlo based paraphrase generation (Chevelu et al)

Chevelu et al propose an approach of paraphrase generation which is based on a Monte-Carlo algorithm[8]. Regarding SMT decoding, Chevelu et al criticise that paraphrases are built step by step following the entries of the paraphrase table. The relevance of steps should be evaluated according to the global paraphrase model. Nevertheless, the computed score depends on the taken path in the decoding lattice. Hence, different paths may produce the same paraphrase but get scored differently. SMT decoding also needs to handle exponentially many solutions. Heuristics like beam-search help to reduce time complexity but do not certainly deliver an optimum. In fact, it is not known whether the true a-Posteriori score will be returned. This might lead to wrong n-best outputs as well. Chevelu et al suggest to regard paraphrase generation rather as an exploration problem than translation task.

The search space consists of *states* which are connected by *actions*. A state contains a sentence and a set of possible actions. An action is a phrase replacement with the position where the replacement takes place. State transformation is done by applying one action of the set of possible actions on the sentence and removing all actions which are not applicable any more. All but the root state belong to the set of final states. An action is not applicable any more if it would modify some phrase which already has been modified by another action. Because sentences are evaluated after complete and not intermediate transformation, computing paraphrases becomes feasable. The sequence or order of transformation is replaced by an unordered set of transformation.
The Monte-Carlo based paraphrase generation (MCPG) of Chevelu et al derives from *Upper Confidence bound applied to Tree* algorithm (UCT). UCT is a Monte-Carlo planning algorithm, basically a search tree and brings along some features:

- The search tree is expanded non-uniformly. Without pruning branches, the most promising sequence is favoured.
- High branching factors can be dealt with.
- At any time, even on interruption, the so-far best solution can be returned.
- State evaluation does not require expert domain knowledge.

Monte-Carlo tree search usually has four phases which are iterated for some time. The amount of iterations can be a fixed value or some function depending on external or internal parameters. Often, the search finishes (and so do the iterations) as soon as some final state has been reached. The phases in the applied order are:

1. **Selection**: From the root node $\mathcal{R}$ follow recursively nodes which have best score until some leaf node $\mathcal{L}$ has been reached.
2. **Expansion**: If iterations should not end with node $\mathcal{L}$ then create at least one child node and from the set of child nodes, select one node $\mathcal{C}$.
3. **Simulation**: Compute or simulate consequences for going for node $\mathcal{C}$ which eventually returns a score value for this particular decision.
4. **Backpropagation**: With the score of node $\mathcal{C}$ in mind, update nodes from $\mathcal{L}$ up to $\mathcal{R}$.

In the selection phase, the most promising path should be taken. Since only a few states are evaluated, the selection process has to decide between areas of nodes which have not been so much explored and areas of nodes which have been thoroughly visited. It is basically a question of exploration or exploitation. An upper confidence bound can be formulated with the estimated value v_i of some node i, the number of times n_i has been visited, the number of times N its parent nodes have been visited and a bias parameter C.

$$\mathrm{UCB}(i) = v_i + C\sqrt{\frac{\ln(N)}{n_i}} \tag{2.48}$$

Extending the UCB algorithm to minimax tree search results in the *Upper Confidence bound applied to Tree* (UCT) which Chevelu et al build upon. A minimax search tree looks for a path of minimal length to maximise the estimated outcome or score. MCPG of Chevelu et al incremently builds the paraphrase.
The root node virtually shifts down in the tree leaving a path of confirmed actions. Each time a new confirmed action is selected, several episodes are sampled. An episode is a path from the current root node to some final state, that is a partially generated paraphrase. The episode is constructed until a stop rule is drawn. The next action is selected depending on whether the state which the episode is currently leading to, has been visited before. If so, the next action is selected according to the exploration-exploitation estimation UCB. If the state has not been visited, its score will be estimated with Monte-Carlo sampling. The episode construction will be finished and all state-action pairs of the episode will be updated. If enough episode constructions are done, a new root state is selected from the child nodes of the current root node considering the maximum score. Then sampling starts off again.
Disregarding the order of replacements and constraining that no segments which already have been replacement might be replaced a second time, delivers an efficiently computable procedure to obtain a true a-Posteriori probability of whether a sentence is a paraphrase or not. In their evaluation, Chevelu et al first test the output of a SMT decoder (MOSES) with the true a-Posteriori procedure. The used models in the decoder are n-gram language model with back-off (SRLIM tool with default parameters and order of 5) and a paraphrase table built with pivoting according to Bannard and Callison-Burch[2]. In order to decrease the size of the paraphrase table, three heuristics are applied:

- *Probability threshold*: Entries with a probability lower than some threshold ϵ are filtered out to decrease noise of coincidences or misspelled words.
- *Pivot cluster threshold*: Large clusters of pivot phrases, that is alignments with some specific pivot phrase might be caused by ambiguous phrases. Thus, pivot clusters which exceed some threshold τ are removed.
- *N-best list*: For computing efficiency, the κ most probable paraphrases for each source phrase are taken.

Chevelu et al empirically set the heuristic parameters to $\epsilon = 10^{-5}$, $\tau = 200$ and $\kappa = 20$. Weight parameters in the log-linear model of the MOSES decoder are usually tuned on a validation set. Chevelu et al argue that for paraphrasing tasks no such validation set exist and hence tuning becomes redundant. The four basic models in MOSES are weighted such that both translation and language model are assigned the same weight ($\alpha_{\mathrm{LM}} = 1$, $\alpha_{\Phi} = 1$), no reordering is allowed ($\alpha_{\mathrm{D}} = 10$) and no specific sentence length is preferred ($\alpha_{\mathrm{W}} = 0$). The training corpus is taken from Europarl with French as the targeted paraphrasing language and English as the pivot language. The test corpus consists of 100 randomly drawn sentences. The MOSES decoder produces a 100-best distinct paraphrases list which is reranked by means of the true a-Posteriori procedure. Both rankings are

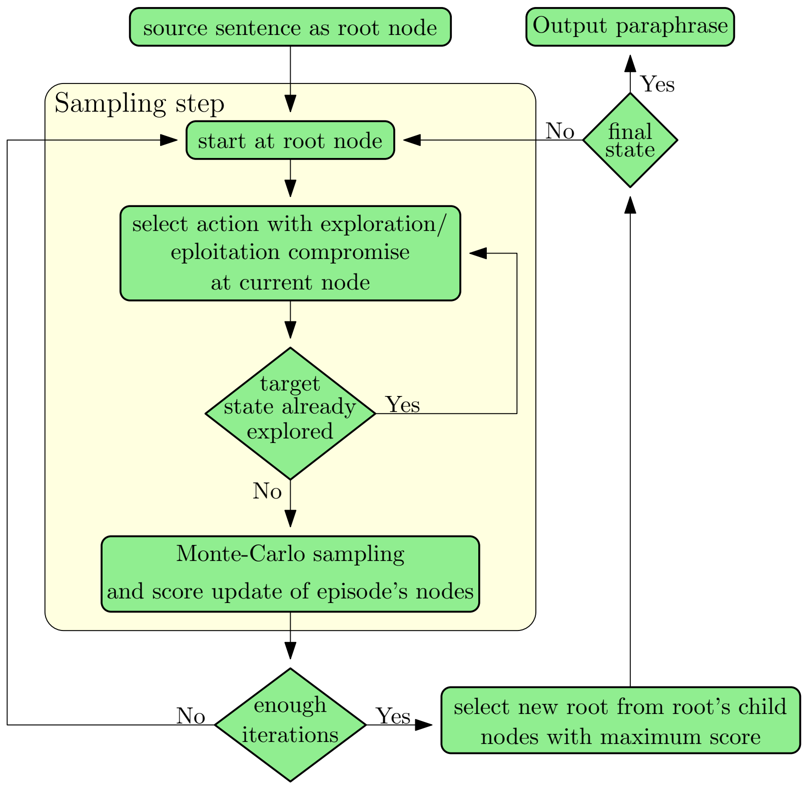
source sentence as root node
Output paraphrase
Yes
Sampling step
start at root node
No
final state
select action with exploration/ eploitation compromise at current node
target state already explored
Yes
No
Monte-Carlo sampling and score update of episode's nodes
No
enough iterations
Yes
select new root from root's child nodes with maximum score

Figure 2.18: Chevelu et al apply the MPCG algorithm on paraphrase generation by successively replacing phrases with synonyms. Partially paraphrased sentences are scored with a language model to identify the most likely sentence

consequently compared with Kendall rank correlation coefficient τ_A. It considers each pair of paraphrases in the n-best list and evaluates how many pairs have kept their relative order (n_p) and how many have changed their relative order (n_i).

$$\tau_A = \frac{n_p - n_i}{\frac{1}{2}n(n-1)} \tag{2.49}$$

With 5-best, MOSES gets the worst correlation of 0.73 which indicates different output but no clear decorrelation. Increasing the size of the n-best list gives better but only modest correlation with about 0.85 at the 100-best list. All in all, MOSES decoder shows to have no strong correlation to the true a-Posteriori probability. In a second step, Chevelu et al take the same paraphrase table and corpora as used in the SMT test. The 1-best output of MOSES reranked by the true a-Posteriori function and MCPG in translator and true-score modes are compared. MCPG in translator mode means that transformations or actions are applied until the whole sentence is covered. MCPG in true-score mode shows that the transformation process can be stopped early with phrases of the sentence not covered. MCPG in this test carries out 100k iterations. Reranked MOSES output gives best results, but it is approximated by MCPG in true-score mode. MCPG in translator mode performs worst. The second test gives evidence that MCPG can reach the performance of SMT systems which are reevaluated whilst working more efficiently due to its reduced exploration space.

3. Methodology

Moore and Lewis[23] give evidence that smaller but more specific domains perform better in Statistical Machine Translation (SMT) than bigger but more general data. Their suggested approach takes use of small in-domain corpus and extracts from a general corpus on sentence level data which is closely related based on some n-gram language model. This idea comes in hand for situations in SMT where the domain of application is too specific to be covered by general corpora and resources to fully describe the application domain are scarce. Employing the small in-domain data set in the approach of Moore and Lewis should allow to harvest closely related data from some big general corpus, bringing along better performance than using the general corpus itself.
In this thesis, several approaches are carried out to examine whether the presented technique of Moore and Lewis can be extended to exploit semantic relations. Synonyms and paraphrases can be seen as fuzzy equality relations. Though paraphrase replacements do not entirely preserve the full meaning, the variations in semantics are low. Moore and Lewis' approach employs an n-gram language model and therefore restricts itself to syntactic patterns. These syntactic patterns which are the n-grams surely carry semantic information but do not exploit semantic information. With the help of additional paraphrase and synonym informations, the small in-domain corpus can be extended. The resulting corpus will be close in its semantic content because paraphrases and synonyms will change semantics only slightly. However, more syntactic patterns can be harvested for selection. Crucial parts are the quality and sensitivity of the semantic information, that is paraphrasing models like synonym tables, and of applying these semantic information in gaining new sentences.

3.1 Synonym Lexicon

Two approaches for semantic analysis are presented. Alignment over pivot language requires a bilingual corpus and exploit indirect alignments. Recursive autoencoder performs a mapping into a semantic vector space and builds upon monolingual data.

3.1.1 Alignment over Pivot Language

Following Bannard and Callison-Burch[2], an approach based on SMT is taken. An issue is to get hands on parallel corpora in order to build an appropriate translation model. The language model can be built in a normal fashion because it requires only a monolingual dataset. Several methods are suggested to extract paraphrases from bilingual or monolingual corpora with SMT tools. It eventually results in a paraphrase table. Our intention is to exploit the pivoting method of Bannard and Callison-Burch to identify synonyms.

With Giza++, an alignment over a bilingual corpus is constructed. The resulting phrase table is marginalised over the pivot language. So two phrases in the non-pivot language, which are aligned to the same phrase in the pivot language, are assumed to be semantically similar, aka synonyms, and hence will be connected. Any alignments which appear less than three times are ignored. Also, alignments whose phrases contain anything else than alphabetical characters or whitespace are removed before computing the margins. The score gives a normalised similarity measure which can be used as joint probability of being synonyms. Each entry then is assigned additional scores, besides the joint probability:

- Levenshtein ratio
- conditional probability for second phrase, assuming the first one
- conditional probability for first phrase, assuming the second one
- difference between these conditional probabilities
- word coverage ratio

The Levensthein ratio r_L allows to identify pair of phrases that differ only in few positions.

$$r_L(a,b) = \frac{\text{Levenshtein-distance}(a,b)}{\max(\text{length}(a), \text{length}(b))} \tag{3.1}$$

The conditional probabilities give insight into how strong synonyms are for both phrases. The difference is a measure on the symmetry between the conditional probabilities. Word coverage ratio r_{wc} considers, how many words are shared regardless of the ordering.

$$r_{wc}(a,b) = \frac{\sum_{w \in \text{vocabulary}(a,b)} \text{abs}(\text{appearance}(a) - \text{appearance}(b))}{\max(\text{length}(\text{words}(a)), \text{length}(\text{words}(b)))} \tag{3.2}$$

Lastly, entries are filterd, so that source phrases have to be in the domain-specific corpus and target phrases have to be in the non-domain specific corpus and additionally not in the domain-specific corpus. After applying all thresholds, a translation table is left. Both conditional probabilities are considered as features.

3.1.2 Recursive Autoencoder

The last approach which will be undertaken, is to exploit recursive autoencoders (RAE) in order to determine paraphrases and therefore to obtain paraphrase lexica of good quality. The idea is to assign semantic representations for phrases with encoding and then to cluster these instances based on their semantic representations.

Because Socher et al report to have gained better results in their RAE application with a grammatical parser [28], the heuristic topology technique is completely omitted. First monolingual corpora are preprocessed via tokeniser and true-caser script from the Moses toolkit. Additionally, any German umlaute and special symbols are escaped, since the parser will not handle such input correctly. We use the same grammar parser, Socher et al uses which is the Stanford parser[1]. The TED corpus is split to be used as test and training data set. The other corpora are EPPS and NC. The amount of sentences increases during parsing because the parser will split some of the sentences. The increase, however is small by tendency. The parser applies a length limitation to filter out any sentence whose length

[1]http://nlp.stanford.edu/software/lex-parser.shtml

Corpus	input	50 cut	100 cut	300 cut
TED test	1500	1675	1700	1701
TED train	168179	187813	191684	191778
NC	201288	200664	208932	209079
EPPS	1920209	1983259	-	-

Table 3.1: Size of corpora in lines in raw (input) and parsed by the Stanford parser with sentence limitations of size 50, 100 and 300 words.

exceeds a certain threshold.
A RAE also requires a code book which maps words into a semantic vector space. The basic corpus for obtaining this code book is built by concatenating EPPS, TED and NC. As before, the total corpus is preprocessed with a tokeniser and a true-caser from the Moses toolset. Like Socher et al[28], we employ *word2vec* for word-embedding. The tool *word2vec* filters any word whose frequency does not exceed a given amount. It aims on removing mispelled or rare words. Some normalisations are applied to bring down non-semantic variations. First, numbers are replaced by some special tag. Second, out-of-vocabulary words (OOVs) are replaced by some other special tag. The idea is to get a special semantic representation for unknown words. For both, the numbers-normalised and the solely-preprocessed corpora, vocabulary histograms are drawn. OOV-mapping then is performed for frequencies beneath 1 (that is in fact empty vocabulary), 3 and 5. All-in-all, eight different code books are built.
The topologied TED test and training corpora are normalised accordingly.

Training is undertaken by grouping the respective topology corpus (regular, number normalisation and oov normalisation) together and varying the semantic vector space dimension, the oov threshold and the cut length for topologied sentences. Besides that, five different combination of activation functions are tried. These are the:

- identity function on both encoding and decoding, making the autoencoder purely linear
- logistic sigmoid on encoding and identiy function on decoding
- logistic sigmoid on both encoding and decoding
- hyperbolic tangent on encoding and identity function on decoding
- hyperbolic tangent on encoding and decoding

In a first run, all combinations for codes of dimension 100 are trained. Using logistic sigmoid as the activation function in both combinations always results distinctively in the worst reconstruction errors compared to the other activation functions. The hyperbolic tangent function sometimes reaches the performance of the identity function. Enabling linear decoding or not shows no big difference.
Adding words with very low frequency (at least one) sharpens the sensitivity but might make thesystem endangered to noisy input. Thresholding the code book to frequency of 3 or 5 appears to make no difference.
It is also not clear whether normalisation delivers any improvement. It seems that the code book already arranges the code instances in a way which makes normalisation obsolent. Normalisation actually might confuse the system due to its artifical nature.
Differences between using maximum sentence length of 300 and 100 are small which is due to the fact that there are only few sentence longer than 100 words and hence, using these as well will not have such an impact on the end result. Using only sentences with at most 50 words however helps to lower the reconstruction errors to a certain extend. Having

min	length	activation	none	numbers	oov	numbers + oov
1	50	id	379.594	435.377	435.797	456.301
1	50	lsigmoid	668.52	637.034	618.902	749.864
1	50	lsigmoid (linear)	479.534	475.753	471.187	497.981
1	50	tanh	439.784	449.719	435.010	463.735
1	50	tanh (linear)	440.396	451.873	432.722	462.207
1	100	id	472.604	475.764	468.901	471.897
1	100	lsigmoid	716.813	694.693	674.291	795.074
1	100	lsigmoid (linear)	519.576	514.068	511.527	539.433
1	100	tanh	480.693	485.626	473.496	503.052
1	100	tanh (linear)	478.506	490.601	475.361	502.473
1	300	id	441.453	468.156	473.025	493.863
1	300	lsigmoid	699.228	697.898	672.009	838.851
1	300	lsigmoid (linear)	521.876	517.664	514.758	541.531
1	300	tanh	480.273	491.605	487.478	505.055
1	300	tanh (linear)	482.382	486.353	476.474	506.114
3	50	id	450.992	426.722	443.235	424.468
3	50	lsigmoid	710.169	691.515	663.007	687.999
3	50	lsigmoid (linear)	492.861	491.620	490.391	503.164
3	50	tanh	456.520	455.296	452.228	463.507
3	50	tanh (linear)	456.592	456.809	453.749	461.396
3	100	id	475.696	491.690	459.191	471.188
3	100	lsigmoid	754.850	732.664	731.109	737.663
3	100	lsigmoid (linear)	533.862	535.297	534.113	545.278
3	100	tanh	494.015	498.998	500.602	505.789
3	100	tanh (linear)	497.645	493.637	493.529	509.388
3	300	id	494.656	486.480	487.093	4.6647e-14
3	300	lsigmoid	760.372	742.777	685.974	7.24753e-14
3	300	lsigmoid (linear)	536.636	538.054	536.179	4.97145e-14
3	300	tanh	503.249	499.694	500.213	5.33157e-14
3	300	tanh (linear)	492.954	497.400	501.665	4.76862e-14
5	50	id	437.275	392.708	433.034	426.571
5	50	lsigmoid	671.787	674.920	687.753	715.170
5	50	lsigmoid (linear)	494.425	490.194	493.358	487.401
5	50	tanh	457.728	451.806	455.255	455.734
5	50	tanh (linear)	458.210	451.295	453.039	454.950
5	100	id	485.604	481.938	475.330	468.741
5	100	lsigmoid	730.144	733.383	740.096	752.486
5	100	lsigmoid (linear)	537.519	530.173	530.902	528.202
5	100	tanh	495.274	494.902	486.637	487.785
5	100	tanh (linear)	497.513	490.339	498.721	490.866
5	300	id	493.470	490.639	4.65301e-14	484.883
5	300	lsigmoid	717.856	7.17853e-14	7.16708e-14	751.192
5	300	lsigmoid (linear)	538.615	5.21704e-14	5.19634e-14	531.765
5	300	tanh	500.07	5.08646e-14	5.12652e-14	496.545
5	300	tanh (linear)	503.971	5.12016e-14	4.69358e-14	490.374

Table 3.2: Minimal reconstruction errors of trained recursive autoencoders; min is the minimal allowed frequency for words; length indicates the maximum length used sentences, activation explains the used activation function where linear indicates an identity activation in the decoding part

Dimension	max length	min freq	error (50)	error (100)	error (300)
100	50	1	379.594	415.762	418.324
100	100	1	434.213	472.604	475.175
100	50	5	437.274	476.984	479.295
100	100	5	446.000	485.604	487.865
300	50	1	1313.38	1429.14	1436.59
300	100	1	1307.76	1422.59	1430.00
300	50	5	1335.70	1453.17	1459.73
300	100	5	1325.51	1442.68	1449.20
500	50	1	2145.85	2334.71	2346.66
500	100	1	2155.25	2344.38	2356.46
500	50	5	2186.79	2379.52	2390.42
500	100	5	2181.63	2374.31	2385.14

Table 3.3: Minimal reconstruction errors on choosen configurations for recursive autoencoders on TED test with cutting length of 50, 100 and 300 words

less words to reconstruct of course keeps in general the sum of reconstruction errors lower. The amount of sentences of size between 50 and 100 however are only 1.5% for test and 2% for training compared to the amount of sentences with length lower 50 words. One potential explanation is that it is much more difficult to correctly find representations for grammatical entities which span over more words.
Proceeding the tests, additional models with dimensions of 300 and 500 are trained with following configuration:

- **dimensions**: 100, 300, 500
- **minimal word frequency in code book**: 1, 5
- **maximal sentence length**: 50, 100
- **activation**: identity
- **normalisation**: none

To see the influence of employing models on data with differently cutted sentence length, all configurations are tested with TED test cutted to 50, 100 and 300 words.
As seen in 3.3, the differences between the different test sets are constant considering each code dimension separately. Dividing these differences by the respective code dimensions gives approximatley the same figures. It indicates that training settings do not prefer cutting lenghts of test data.
Increasing the minimum frequency of code words worsen the performances. Since the model works on such a fine-grained level in terms of semantics it is essential to have as few unknown words as possible.
The cutting length of training corpus works in relation with the code dimension on the performance. High dimension of the semantic space allows to encode more information. But too much information can oversaturate the model and decrease its quality. For a dimension of 100, it appears that latter is the case. With higher cutting length in the training set, test results get worse. Taking a semantic space of dimension 300, longer training sentences actually increase performances on the test corpora. A semantic space of dimension 500 tends to overfit. Employing a code book with a minimum frequency of 1 leads to a decrease in the performance when using longer training sentences. Having a minimum frequency of 5 in the code book results in a performance increase when using longer training sentences.

To draw conclusion from the preliminary test runs, the configuration with dimension 100, cutting-length 50 and minimum frequency 1 is taken. Configurations with dimension 300 and 500 are dismissed due to their time consumption.

- Dimension: 100, Cutting-length: 50, Minimum frequency: 1
- Dimension: 300, Cutting-length: 100, Minimum frequency: 1
- Dimension: 500, Cutting-length: 100, Minimum frequency: 1

Monolingual corpora are emploid to these three models with cutting-length 50. The output is a list of phrases and their code representation in the semantic space. For all phrases which appear as well in the domain-specific corpus, the 50 closest neighbours and respective distances are computed. To apply normalisation, the sum of all distances for target sides is required as well. For this reasons, sum of distances to 50 closest neighbours for all phrases on the target sides are computed too.
Each pair of neighbours is assigned the same metrices as done before for the n-gram pivot alignment. Normalisation is performed with distance t_1 and t_2 to the 50-closest neighbours. Then, normalised distances d are transformed to normalised similarities s_1 and s_2.

$$s_i = \frac{t_i - d}{t_i * 49} \tag{3.3}$$

Subsequently, thresholds for metrices are applied. Any pair whose target side does not appear in the non domain-specific corpus is removed. Any pair whose target side does appear in the domain-specific corpus is removed too.

3.2 Paraphrasing

Two methods are applied in order to enrich the domain-specific corpus with information of synonyms. The first method is a simple random replacement. And the second method follows a SMT based approach.

3.2.1 Random Replacement

Random replacement takes a synonym lexicon, a number indicating how many iterations of replacements per input sentence should be carried out, and a parameter value for a probability distribution which tells whether some word should be replaced or not. Latter value is used as parameter in a Bernoulli distribution. The outcome of a Bernoulli distribution is dual, that is $\{0, 1\}$ or $\{true, false\}$. In the implementation of random replacement, 1 (or $true$) is understood as performing the replacement whilst 0 (or $false$) will keep the respective word or phrase untouched. Basically, drawing a sample from the Bernoulli distribution delivers an answer to the question if replacement should be carried out or not. A parameter value close to zero will cause only few replacements and a parameter value close to one will result in many replacements. To allow for replacements of phrases and not only words, the sentence is scanned in a forward pass and each time, possible paraphrase candidates are found, the Bernoulli distribution is drawn for replacement decision. If this decision is yes, one paraphrase will be drawn randomly from all possible paraphrase candidates with a uniform distribution. This paraphrase will be written to the output and the scan will continue behind the position to which the paraphrased phrase reaches. If no replacement is undertaken, the scanned word will be printed to the output and the scan will continue with the next word. No language model is applied and neither are probabilities of paraphrases considered in this random replacement algorithm. The outcome is not correct in terms of syntax but should extend the sentences semantically, depending on the quality of the lexicon. With the risk of uncorrect output, noise is introduced which might have an

Algorithm 1 Random replacement: argument l is the index of the left most leaf index in the subtree, r is the index of the root node in the subtree and f defines an action for nodes

```
Function replaceRandomly (lexicon, nIteration, param):
for all sentence in input do
  words = split sentence into words
  // paraphrases is array of (startIndex, endIndex, target)
  paraphrases = lexicon.lookup(words)
  for cIteration = 1 to nIteration do
    // ip is index of array paraphrases
    ip = 0
    // i is index of array words
    i = 0
    while i < #words do
      if ip ≥ #paraphrases or paraphrases[ip].startIndex > i then
        write words[i] to output
        i = i + 1
      else if bernoulli(param) is yes then
        q = #(filter paraphrases by startIndex equals i)
        j = randomly drawn from uniform distribution in [0 ... q-1]
        write paraphrases[j].target to output
        i = paraphrases[j].endIndex + 1
        ip = find first index in paraphrases[ip ...] such that startIndex > i
      else
        write words[i] to output
        i = i + 1
        ip = find first index in paraphrases[ip ...] such that startIndex > i
      end if
    end while
  end for
end for
```

effect on the eventual translation performance. For selecting one paraphrase from the set of possible paraphrase candidates, a uniform distribution is choosen because paraphrase candidates may comprise different source phrases which only share the first word. Without language model, it is difficult to give appropriate probability proportions. So, uniform distribution is choosen due to simplicity. The domain-specific corpus is fed to the random replacement algorithm with different Bernoulli pararameters ten times. Subsequently, the extended corpora, that is the domain-specific corpus concatenated with the newly formed sentences, are freed from duplicates.

3.2.2 SMT

Quirk et al show that an SMT system can be used to carry out translation within one language rather than translation between two languages[27]. The idea is that rather than allowing randomness to decide which phrase to replace, a score will indicate which new sentence will be the most appropriate. The translation model puts contraints to firstly which phrase can be potentially replaced by which other phrases and secondly, how semantically close these replaced phrases are. The language model evaluates a sentence according to its correct syntax. Assessing each possible combination will lead to exponentially many branches and therefore is not feasable. It is the same problem, SMT is faced with. Heuristic pruning, also called *decoding* in SMT, helps to bring complexity down to linearity. The modern standard of SMT decoding is a log-linear model of features like translation or language model. Each feature is assigned a weight which represents the impact of the feature in the final score. These weights are determined on a held-out corpus to get best results. In our scenario, the translation model translates synonyms within one language. Like in the random replacement approach, the domain-specific corpus is fed to the decoder and the 20 best translations are put out. Subsequently, the extended corpus, that is domain-specific corpus and the 20-best translations, are concatenated and freed from duplicates.

3.3 Text extraction from PDFs

Many reports and theses are packed as PDF (Portable Document Format). Not only is it the standard text document format on the internet. It also is supported in various document editor frameworks such as Microsoft Word, Open Office or pdflatex.

The design of PDF aims to obtain a device-independent representation of enriched text documents, that is text documents with optional graphics and images as well as format information. PDF itself uses PostScript, a stack based language which is Turing complete. PostScript is developed by Adobe for describing document pages in vector format. Exploiting the nature of vector graphics allows to find a graphical visualisation independent on display devices. But contrary to HTML, no logical structure of the document is necessarily stored. This fact makes it rather difficult to retrieve the logical relations between different items in the document. Hence, any text extraction will be noisy.
Some tools are presented and two heuristic approaches are proposed, one for general text extraction and another for abstract extractions.

3.3.1 PDF format

PDF is designed to display documents in a device-independent fashion. Different specifications and extensions exist. Yet three components are in place.
A general system allows to store objects with associated content which - depending on certain factors - will be compressed. Second, fonts can be stored as well to not rely only

on fonts given in the specification. The third component is a subset of the PostScript language. With loop and decision controls as well as commands, the document is described which allows for rescaling and any pixel-based post-processing without further information.

Since the focus is on visualising the document, any logical information will be omitted. Some specification permit PDFs to store the logical structure which is refered to as tagged PDF. Elements are hierarchically connected allowing to identify logical entities. PDF producing tools nowaday store the logical hierarchy of the document. Nevertheless, it cannot be assumed to be used everywhere for reason of parameter settings or modifier tools which may omit logical informations for reasons of complexity and ambiguity.

3.3.2 Tools

Several tools for extracting information out of a PDF already exist. Usually, these tools are designed to convert or filter out informations.

3.3.2.1 XPDF

A widespread tool set is XPDF [2] which contains open source tools to view and filter PDFs. The original aim is to provide a graphical viewer for PDFs called *xpdf*. Since PDF is based on PostScript (PS), there is also a PDF to PS converter (*pdftops*). Additional tools allow to extract images, fonts, attachements or meta informations. Besides to its bitmap converter, XPDF offers a PDF to text converter (*pdftotext*). The PDF to text converter displays the textual content of the PDF as precise as possible onto a text file. That, however, does not deliver an appropriate output for text processing. Text in columns is still displayed in columns and if for some reason lines of two parallel columns differ more than slightly, an empty line will be alternately inserted in both columns.

3.3.2.2 Popple

An extension of XPDF is Popple [3]. It not only contains alternative implementations of the XPDF tool set. The tool *pdftotext* supports different outputs like raw output, that is strings appear in the order they appear in the document, with bounding boxes or in html format. Another tool, *pdftohtml*, offers similar features. *pdftohtml* converts a PDF into an HTML formated document. Besides basic html formating, it also can print out the text content in XML format, including information about bounding boxes formated document. Besides basic html formating, it also can print out the of pages and text snippets. formated document. Besides basic html formating, it also can print out the

3.3.2.3 PDFExtract

Besides simple tools for working with PDF, there are some approaches which aim towards extracting textual content from more complex PDF documents. In particular, scholar documents have been the target of a text extraction application developed by Raddum Berg et al [5]. The source code is publicly available[4].

Raddum Bert et al reconstruct the logical structure of a PDF with techniques similarly used in *Optical Character Recognition* (OCR). The logical structure is derived from a layout analysis which consists of two consecutive processes: geometric layout analysis which groups elements in the document together according to their geometric properties, and logical layout analysis which determines what belongs to the text flow and what is additional information.

[2] http://www.foolabs.com/xpdf/
[3] http://poppler.freedesktop.org/
[4] https://github.com/CrossRef/pdfextract

The analysis starts off by finding whitespace delimiters. That is empty rectangles which are maximed over their space. A set of rectangles is held, initialised with a rectangle covering the whole page. As long as some rectangle exists in this set which is not empty, that is it contains some bounding box, the iteration of whitespace delimitation does not stop. Each rectangle which is not empty, will be replaced by four sub rectangles. Therefore a pivot bounding box which lies within the rectangle is selected and the sub rectangles above, underneath, left and right from the pivot bounding box are take instead. With some heuristics, small issues in quality and performance are adressed. In particular, whitespace rectangles within contiguous text segments and small whitespace rectangles between lines within some paragraph need to be avoided.

Next, geometrical layout analysis is performed. The set of whitespace rectangles is taken

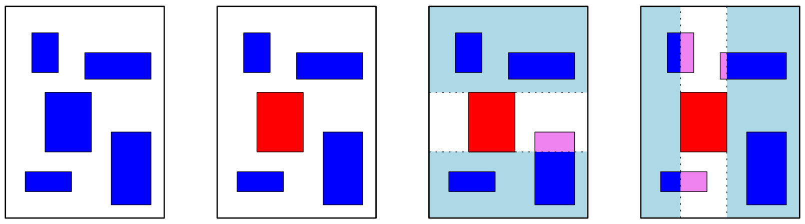

Figure 3.1: Illustration of one iteration in the whitespace covering algorithm: as long as a rectangle contains any bounding box (blue boxes), it will be recursively separated into four rectangles with a pivot bounding box (red box).

in order to find *blocks* of homogenous content. First, column boundaries are detected in a three-step approach, whilst erroneous whitespace rectangles need to be dealt with:

1. extract initial set of candidate boundaries
2. heuristically expand column boundary candidates vertically
3. combine logically equivalent boundaries and filter unwarranted boundaries

Second, non-whitespace elements are grouped into *blocks* which ideally are paragraphs, headings, footings and so on. Adjacent and non whitespace elements are grouped together if no intervening whitespace rectangle divides these elements. Mathematical equations which are identified by content or font properties, are treated differently to avoid them causing a block separation. Horizontally oriented whitespace rectangles around mathematical equations are ignored to some amount. At last, the reading order is recovered, that is which block succeeds which other block. The recovery is based on topological sorting of lines with help of relation of hierarchical nesting and relative geometric positions.

Geometric analysis highly depends on accurate coordinate information of glyphes used in the documents. For several reasons, however, these information become incorrect. These are variations in font types, missing informations for embedded fonts and bugs in *PDFBox* which is the underlying PDF library used in *PDFExtract*. Most of the problems can be resolved with patching *PDFBox* and special handling of unknown fonts. Besides font handling, Raddum Berg et al identify another issue which is word segmentation. Whitespaces are used between words to separate them as well within a word to separate its characters. Spaces between words are distinctively longer than in between words. Yet both kind of spacing can vary frequently. To figure out whether spacing is set to separate two words or just two characters within a word, Raddum Berg et al average a selection of small character distances within a line. This average distance is compared to all character distances within the line to segment the characters into words.

Logical layout analysis tries to reconstruct the logical hierarchy with the blocks coming from the geometrical layout analysis. First, a set of text styles is inferred, that is unique combinations of formating properties such as font type and size. Then, different components are identified with heuristic rules:

- **Body text**: In terms of number of characters, take the most frequent text style.
- **Title**: Take the text style from the header-like text blocks on the first page which has the largest font size.
- **Abstract**: If on the first page some single line text block contains the word "abstract" and has a bigger or bolder text style than the the body text, its text style will be taken for the abstract and all body text until the next heading is added to the abstract.
- **Footnote**: Text blocks placed in the lower part of the page are searched for starting with a number or some footnote indication symbol and for smaller font size than the text style of the body text.
- **Sections**: A list of style is compiled such that it contains styles which are larger than the body text or contain an emphasis on the body text. Text blocks having these text style are tested on initiative enumeration instances. All remaining text blocks are seen as section headers. The nesting level of the sections is inferred from the order of occurences.

The text blocks which are assigned to the body text are merged such that merging text blocks are consecutive and have identical styles. From there, paragraphs are formed based on indented initial lines. Eventually, dehypenation is carried out with a lexicon and a set of orthographic rules.

3.3.3 Heuristic text extraction

Based on bounding box and formation style information of text snippets in a PDF, a heuristic approach can be taken to derive related text snippets. It follows a bottom-up process in which related text entries are identified and merged. Consequently, no direct logical structure is produced. Rather, text snippets are grouped to ensure that they belong together. For Statistical Machine Translation (SMT) or Natural Language Processing (NLP), text snippets should work fine, since both SMT and NLP only need a limited context information to infer statistical models.

Bounding box and formation style information can be obtained by tools like *pdftotext* or *pdftohtml* from the Poppler tool set. In fact, for experiments *pdftohtml* with XML output and no paragraph merging is used[5]. On the way to reconstructing the logical relations between text boxes, some problems arise:

- **Page break**: The xml input is partitioned into pages which induces page breaks. However, page breaks, of course, can introduce logical entities like new chapters. But page breaks appear as well within chapters if a page is fully exhausted. Sentences or even words may be split by page breaks. A top-down approach should consider neighbouring pages to figure out where the text flow continues on the next page.
  In Western languages one usually can assume that the text flow starts in the top left corner and ends in the bottom right corner. Lines are read from left to right and multiple lines are read top down. In some languages the reading/writing order is different. Japanese knows a traditional way of reading/writing which is column-wise top down and going from the left column to the right column. Japanese has a modern way of reading/writing which is the same as Western languages like English do. In our bottom up approach, we assume mostly Western languages such as German,

[5]pdftohtml -xml -noframes -stdout -nomerge -enc UTF8 -nodrm

French or English. Thus, the direction of reading/writing is believed to be line-wise left to right and lines are read top down. A more sophisticated system might check upon characters to identify the correct writing/reading direction. In our implementation, pages are processed individually, that is we do not handle page breaks but go without extended context and hopefully see less noise. Finding the entry group of text boxes on the next page will require considering different features like formating style, position patterns, patterns of annotation and a language model. Annotations are covered in the next point. The language model is designed to test the most convincing continuation in thze text flow over the page break. Some non continuous text elements like graphics, tables, page annotations or titles will make it more difficult to find the right entry point.

- **Page related annotations**: Almost every document annotates its pages in some way. Often, the page numbers are placed at some specific position. The current chapter name or copyright information can be found at the right, left, upper or lower margin. Foot notes, as the name suggests, give extra information and are placed in the lower margin. Additionally, some superscript number is inserted into the text. One can also find temporary annotations at the left or right margin which comment on the text at this position. It may contain some small summary, key words or corrections. These annotations typically have regular patterns of formatting style, positions and content. Nevertheless, it is not trivial to identify these annotations because some might appear on almost every page whilst other occur rarely. If the document is composed in book style, variation increases because left and right page in a book are designed differently. Hence every second page might share some patterns in their annotation. Since annotations usually contain little information, it should lower noise, if these annotations were filtered out. It can be achieved by some classification process for which features have to be selected. Another approach is to find a big box on the page which will only contain the actual text and remove all elements outside of this box. A third approach we are using, is to group text boxes and to filter out groups which fulfill certain criteria like containing only numbers.

- **Caption text**: Other potential sources of noise are captions of tables or graphics. They might contain phrases which increase contextual information. The issue lies in the placement of these elements relative to the text. Tools like *pdftotext* or *pdftohtml* only pass on text boxes, any graphical information like images or simple geometric elements such as lines are filtered out. There are few ways left to identify captions. If captions are printed differently, formating style or relative positioning might help to conclude a classification. It is however not given for sure that captions are formated differently. Scanning on key phrases at the beginning is another approach. Often, captions are composed of an identification and a description. The identification starts with some word like T̈able; F̈igure¨ or G̈raphics¨ and continues with an enumeration like Ï.2.a: Combined with formatting style and information about the position, it will give a better decider on whether some group of textboxes is a caption or not. The key phrase is not best because some documents use different words in the identification part or even do not use any identification. The set of key phrases also depends on the language in which the document is written. With the information about graphics, tables and other non-text elements, it should be easier to find possible captions. Going bottom up, situations exist where captions can be missinterpretated as preceding paragraphs. At higher computation cost, language models can be used to find the most likely text flow.

- **Mathematical equations**: In many scientific documents, mathematical equations appear in an emphasized setting. The text flow stops and the equation is printed underneath the text. The equation or list of equations is optionally enumerated at the right or left margin. The text flow then continues underneath the last equation. The equation itself consists of various text boxes which are differently aligned due to fractions, indices, subscript or superscript expressions or symbols which are placed separately. Two characteristics help to detect equations. The first one is the set of math symbols used in the equation. The second characteristic is the distinctive placing in the text flow. It allows to draw a bounding box around the equation which is complete in a sense that the whole equation is inside whilst no element which does not belong to the equation is inside. The semantic value of equations is rather low, so adding it to the corpus might introduce unnecessary noise. Sometimes, the equation is part of a sentence. If the full sentence should be reconstructed, the whole equation might be reduced to some special entity word which indicates an equation. Since we are interested in limited context, the full sentence reconstruction is not important. Therefore, an equation is seen as separation. The equation with all its text boxes is put into its own group which then can be filtered out during the post processing. If the equation is part of a sentence, the prior and posteriour part of the sentence will end up in different groups.

- **Tables**: Tables in a PDF document are built by lines and text boxes. Lines will be filtered out by tools like *pdftotext* or *pdftohtml*. The textual content of a table is therefore arranged in columns and rows. Since tables can be arranged freely, they also produce noise in situations where they are put within some text. The absence of drawn line information makes it more difficult to detect tables in the converted PDF document. Entries in a table might have meaningful text, but often are few words, numbers or abbreviations with small context information. Filtering out table entries should help to keep down noise in the data. Similar to mathematical equations, grouping from bottom-up allows to exclude table entries from the remaining text, because tables are framed and naturally bring along some bounding box. Entries with only few words will be filtered out in post processing. Of course, this method does not directly identify a table as such. To detect tables, more data like information about drawn lines or captions with key words are required. Another way would be to apply a language model which extracts the main text content. All remaining elements then are classified. Arrangements and positions of the remaining elements should allow to find out whether they are tables, page numbers or some other element.

- **Columns and blocks**: Grouping text into columns is a widespread technique for document design. It is easier for the human eye to comprehend at a first glance a line with few words than a line with many words. It is for this reason that text is arranged in columns and not extended to cover the whole page width. In the process of text extraction, the amount of columns have to be determined or somehow taken into consideration. Typical for columns is the shared left boundary on each line (with the optional exception of the first line) and the mostly shared text style as well mostly same or similar distances between lines. Some obstacles can separate a column into two blocks, even though the two blocks belong together. Obstacles can be mathematical equations, tables, pictures or even text like citations. In the spirit of needing only limited context, handling these obstacles can be ignored. This will cause a column to be split in some upper and lower part for which the respective context is limited to. Grouping text in columns is necessary if the context should extend over a line which are aiming for. A new complexity is introduced if a logically different text block is placed partly into a column such the text of the columns flows around the text block. Solving such a situation needs more complex handling.

- **Emphasising formatting**: Formatting is rather a soft than a hard criterion for distinguishing logical entities. In some situations it should be used to decide for having found a new logical entity like in titles whilst in other situations it should be better ignored. Latter often happens in continuous text. Variables are usually set differently. Some greek characters are placed lower than the surrounding text. A popular method for emphasising is to exploit bold or italique fonts. Sometimes, the first character in a section is enlarged to some size. Handling this character as a word will break the actual word and hence will introduce noise in the data. In situations, like this enumerations, the initial expression is emphasised to illustrate that this particular expression is described in more details. Separating the two differently formated text does not introduce noise. A non-wholistic approach such as our implementation should take different features into considerations. Patterns on the initial text as well as relative positioning within the text give better indication on how to proceed with differently formated text.

- **Non semantic characters**: To support document layout, a variety of graphical symbols exist. Primarily, bullet points and enumeration instances are put in place with holding no semantic information and yet being handled as part of the text. In a more extended point of view, mathematical equations, variables and terms can be seen as non semantic characters as well. In a top-down procedure, such characters can help to conclude the kind of logical entity a text block owns. However in our bottom-up approach with limited context interest, these non semantic characters usually do not contribute in any way. Post-processing should remove them from the data. Bullet points can be easily detected and mathematical equations too. Mathematical variables and terms though are not simply distinct from the context. Enumeration instances at the beginning of a line like in titles can be detected with pattern matching. References which are expressed by enumeration instances and are placed within text are much harder to detect. The simplest way to avoid noise caused by these non semantic characters is pattern matching in the post-processing step.

Some of the problems Rannon Berg et al experience, do not appear in our implementation. In particular, *Popple* seems to deliver well formed text snippets which consists of words next to each other with the same text style configuration. Besides font style, the bounding boxes are given so any coordinate infering based on fonts is obsolote. Issues with incorrect coordinates does not appear although some letters, especially greek ones, are put slightly lower. Spacing between words differs slightly too. Considering these variations are incorporated into the extract procedure. Contrary to *PDFBox*, word segmentation is not an issue apart from some seldom cases. As discussed before, we follow a bottom-up approach and do not aim towards fully reconstructing the logical structure. Instead, only text snippets with a context length of some sentences at most is expected. Each page of a document is handled separately, that is no entry point on the next page is identified and no order of text flow is determined. Our procedure rather limits itself to a geometrical layout analysis compared to the solution by Rannon Berg et al. Pages are composed of *levels*, *columns* and *paragraphs*. *Levels* are the most general entity spanning over the document in vertical direction. It should separate headings, body text and footings from each other. In particular, title pages can be correctly segmented with levels. Containing only titles and some information about the author, supervisor and institution, the logical direction usually is from the top going down. Often, the body text already starts on the first page and is placed beneath the title and the name of the authors as well some contact informations and optional sub titles. Assuming a column oriented ordering would create noise. Additional, headings and footings would require to get somehow separated from the body text. Within a *level*, columns are assumed, that is blocks are positioned from the left to the right. In title areas, no two logicaly different text entities are placed horicontally next to each other. No columns in body text actually means exactly one column in the body text. The lowest grouping is *paragraph* and it is vertical segmentation within a *column*.

Paragraphs allow to separate text within a column where titles or some graphic with caption appear. Generally, the separation are infered from relative positioning and expecially spacing, that is the minimal vertical distance between two text blocks. The text extraction which we propose, is built on two basic methods: clustering and merging. Heuristics are subsequently applied. The input is a list of text boxes which contain information about the bounding box, text style and the text itself.

1. **Grouping to lines**: The intention of grouping text boxes to lines is to find coherent lines. The list of text boxes is recursively clustered in horizontal and vertical orientation. The clustering starts with an initial set of text boxes which at the beginning holds all boxes. The set is sorted by the *left* coordinate of the bounding boxes. Then boxes are grouped together if neighbouring instances overlap or closely overlap. For each resulting group, the same procedure is performed on the vertical direction, that is sorting by the *top* coordinate of text boxes and grouping if neighbouring instances overlap. On horizontal clustering, no close overlapping is allowed. The clustering is continued on the resulting groups of the vertical clustering. The clustering stops as soon as no further change happens. Allowing for close overlap in horizontal direction brings closely positioned text boxes together into one cluster. Each cluster then is sorted from the left to the right which is the text flow direction of a regular line. All clusters are reboxed, that is a line cluster is assigned a bounding box with a minimax box resulting from the bounding boxes its text boxes, and the content of the text boxes is concatenated with whitespace in the order of the left-to-right sort.

2. **Grouping to paragraphs**: Paragraphs are built of lines which are direclty underneath each other. Distances between lines may vary. But lines usually hold on to the same indentation. First, associations between tuples of lines are created. The association tells if the second line in the tuple is a potential successor of the first line in the tuple. Each line box has a set of potential succeding line boxes. Initially, all line boxes which are completely lower and have some overlap on the horizontal projection, are considered. Each set is further reduced by removing any line boxes which lie below some other line box in this set. Additionaly, any line box b of a candidate set belonging to some line box c is removed if the vertical distance between c and b is more than $\frac{3}{4}$ of the height of b or c. Eventually tuples are built from each line box and each line box remained in the asssociated set. These tuples are sorted by their vertical distance, smallest first. After that, tuples are grouped together if the next tuple's distance is at most 10% larger than the lowest distance in the current group. The idea is that association between two lines is stronger if the two lines are closer. Ranging the accepted distance upt to 10% accounts for variation in character settings or indices. Subsequently, any tuple of line boxes is removed if the minimum height differs more than 10% of the maximum height and if the lower box is more right than the upper box. Latter requires the upper line to have equal or smaller indentation than the lower box. Finally, tuples are cleaned to have no duplicate line boxes. To do so, groups are filtered (in the order of smallest distance to largest distance) to have only tuples of line boxes which do not appear in any previous group. Tuples are clustered seeing the association pairs as equivalence relation. Any line box which does not appear in any cluster, builds its own cluster. The clusters contain line boxes which have almost identical in-between distances and are placed one underneath the other. These clusters usually build paragraphs and are boxed, that is each cluster has a minimax bounding box and its line boxes as data.

3. **Grouping to columns**: The paragraph boxes are clustered recursively in horizontal and vertical orientation once again to account for fractions and bigger variations in spacing. Contrary to clustering the text boxes, no small gap is tolerated. The newly

created clusters are boxed and form columns.

4. **Grouping to levels**: The last step in reconstructing the geometrical layout is to determine which column boxs are next to each other. It mostly is necessary to distinguish document titles, headings and footings from the body text. First, column boxes are once clustered vertically and then clusters which have more than one column are grouped together in descending order. These groups build the levels.
5. **Text flow order**: The layout is sorted to follow the text flow. Levels are sorted top down, columns are sorted left to right and paragraphs are sorted top down again.

The output actually considers only paragraph level because no entry point has to be found and it fulfills our requirement. Before printing a paragraph, dehyphenation is applied. If longer context is required, the next paragraph has to be determined based on stylistic, geometrical and optionally lexical information.

4. Experiments

For all experiments, a random selection of 40 000 sentences is taken from the TED corpus as domain-specific corpus. It is referred to as *TED in-domain.* For perplexity measurements, a second corpus is created which is close to *TED in-domain.* But it is neither a super nor a sub set. 30 000 senteces are taken randomly from TED in-domain and concatenated with 10 000 sentences randomly drawn from TED. It is referred to as *TED close-in-domain.* Both share 43 748 monograms. TED in-domain has 6 661 unique monograms and TED close-in-domain has 6 485 unique monograms.
Two non domain-specific corpora are taken. A collection of academical papers is collected from the Web. Many German universities operate *Hochschulschriftenservers.* These are servers dedicated to offer theses, reports and any other academical text written at the respective university. The mostly used software for these *Hochschulschriftenservers* is *OPUS*[1]. It is fairly simple to write a script that goes through the directories of a *Hochschulschriftenserver* and searches for any PDF document. A list of *Hochschulschriftenservers* is available on the internet[2]. Subsequently, text snippets are extracted from the PDFs with the method presented in 3.3.3. With three language models for German, French and English, the text snippets are grouped. For this work, only text snippets assigned to German are exploited. It is referred to as *Papers.* The second non domain-specific corpus is a collection of sources taken from Mediani et al [20]. It is referred to as *Collections.*

4.1 Evaluation

Two kinds of evaluation are performed. The intrinsic evaluation measures the perplexity targeting TED close-in-domain. Taking orientation from Mediani et al, limitations will be set on the considered vocabulary. The first limitation is the intersection of vocabulary from selected subcorpora. The second limitation is the union of vocabulary from selected subcorpora. Also, perplexity with no limitation for vocabulary is tested. Different selection sizes are taken: 0.1%, 0.2%, ..., 2.0%, 2.5%, ..., 10% Minimal perplexity on the TED close-in-domain decides which selection size is considered in the evaluation system.
The extrinsic evaluation is BLEU scoring for an English-German translation system. The SMT system in use is taken from the research team of Prof Waibel at Karlsruhe Institute of Technology (KIT)[14]. The metric for comparison is BLEU. It scores translation hypotheses against translation references considering precisions in n-gram model of order

[1]http://elib.uni-stuttgart.de/opus/
[2]http://www.dini.de/dini-zertifikat/liste-der-repositorien/

Corpus	lines
NC	201 288
TED	171 721
EPPS	1 920 209
TED tuning	1 433
TED testing	1 700
Papers	8 289 555
Papers filtered	18 270
TED rest	129 678
TED in-domain	40 000
TED close-in-domain	40 000
Collections	10 428 543

Table 4.1: Corpora used in experiments

four. BLEU is regarded as having high correlation with human judgement. Features are a language model and a translation model. The language models will be varied for different configurations.All language models are n-grams of order four with Knesser-Ney smoothing. The translation model is built from EPPS, TED and NC corpus and aligned with GIZA++. The test and tuning sets are done on small heldout TED corpora.

4.2 Baseline

Eleven different baselines are drawn for further comparison. The first baseline (Baseline1) takes a language model which is trained on EPPS, NC and TED rest corpus which is the TED corpus minus the TED in-domain corpus. The second baseline (Baseline2) uses an additional language model trained on the whole papers corpus. To show the quality of the papers corpus, the third baseline (Baseline3) only uses the language model trained on the Papers corpus. Since the Papers corpus contains a lot of noise, a language model is built upon a filtered version of the Papers corpus. The filter is a threshold (≥ 0.9) of alphabetical characters to all characters and a threshold (≥ 4) upon word counts. The filtered Papers corpus contains 18 270 sentences. That is about 0.22% of the unfiltered Papers corpus. Likewise, baseline configurations are drawn with Collections.
With the TED in-domain corpus, selection according to Moore and Lewis is performed on the papers corpus. Perplexity is tested for various sentence lengths of the final selected corpus. According to Median et al, intersected and unified vocabulary are considered on the analysis on TED close-in-domain. Both vocabularies show a minimal perplexity at 2% with 93.5783 (intersection) and 3% with 151.467 (union). Both show as well the typical curve that is described by Moore and Lewis6.4. Applying the selection by Moore and Lewis on Collections results in a minimal perplexity of 95.4604 at 4% for intersected vocabulary and 221.132 at 7.5% for unified vocabulary6.5.

The selection with minimal perplexity on intersected vocabulary is used as training corpus for another language model which is considered in combination with the language model used in Baseline1 (Selection1, Selection3) and alone (Selection2, Selection4).

Even though the papers corpus is eight times the size of EPPS, NC and TED rest combined, it significantly performs worse due to its noiseness (about 3 BLEU points). The affect of noise in the papers corpus appears as well in Baseline2 configuration where both language models are combined. Slight decrease of 0.36 BLEU points show the raw nature of the papers corpus. Emploing harsh filtering on the Papers corpus removes much noise but not much is left. Baseline4 delivers best results with a BLEU score of 19.61. So, Papers corpus contains some - though few - information related to the test set. Using only the

Configuration	LM1	LM2	LM3	LM4	LM5	LM6	BLEU
Baseline1	x						19.56%
Baseline2	x	x					19.20%
Baseline3		x					16.62%
Baseline4	x			x			19.61%
Baseline5				x			16.25%
Baseline6	x				x		19.53%
Baseline7					x		19.38%
Selection1	x		x				19.13%
Selection2			x				16.54%
Selection3	x					x	20.90%
Selection4						x	19.19%

Table 4.2: Baseline configurations and evaluations; LM1 (EPPS, NC, TED rest corpora), LM2 (Papers corpus), LM3 (Papers selection), LM4 (filtered Papers corpus), LM5 (Collections), LM6 (Collections selection)

filtered Papers corpus for the language model (Baseline5), the worst BLEU score is reached (16.25).
In the selection process, noise is filtered out as a byproduct. It nevertheless does not exceed the performance of only using EPPS, NC and TED rest (0.43 BLEU points difference). This low-performance indicates that the content of selection from papers corpus is not so closely related to the TED test corpus as it would bring any enhancement with Moore and Lewis' approach. Using the selection of papers corpus alone confirms its semantic distance to the TED test corpus with achieving 0.05 BLEU points less than using the whole papers corpus alone.

Approximately equal in size to Papers, Collections has less noise. Its language model can be used alone (Baseline7) and achieves comparable BLEU score to Baseline1. Combining its language model with the language model used in Baseline1 (Baseline6) shows that the information from both sources are vastly equal according to the BLEU score. Collections neither add new information nor does it pollute the information coming from EPPS, NC and TED rest.
Applying the selection alone, that is using its language model alone, gives a slight decrease in the BLEU score. Collections contains abundant information and considering only a part nevertheless returns in a system which performs slightly worse than Baseline1, Baseline6 or Baseline7. Best BLEU score is reached with Selection3, considering a language model trained with EPPS, NC and TED rest and a language model built upon the seleciton.

4.3 Alignment via Pivot Language

The German-English part of EPPS is taken for alignment via pivot language. The pivot language is English, so synonyms are obtained for German. After constructing the alignments via Giza++, following thresholds are applied on the metrices:

- Levenshtein ration: ≥ 0.6
- conditional probability: ≥ 0.001
- word coverage ration: ≥ 0.34

4.3.1 Random Replacement

Applying random replacement results in a similar result for Papers and Collections over all three Bernoulli parameters 0.2, 0.5 and 0.9. Limiting vocabulary to the intersection or union gives the typical curve of perplexity. Using no restrictions on the vocabulary results in a rather flat line 6.6 6.7. Perplexities on intersection and union have respectively similar minima regarding the different distribution parameters. Compared with the perplexities in

	limitation	distr. parameter	min. perplexity	selection size
Papers	intersection	0.2	92.9244	207239 (2.5%)
Papers	intersection	0.5	93.3294	165791 (2.0%)
Papers	intersection	0.9	90.2959	165791 (2.0%)
Papers	union	0.2	158.736	149212 (1.8%)
Papers	union	0.5	159.418	157502 (1.9%)
Papers	union	0.9	159.465	140922 (1.7%)
Collections	intersection	0.2	98.7232	417142 (4.0%)
Collections	intersection	0.5	99.1992	469284 (4.5%)
Collections	intersection	0.9	97.2917	521427 (5.0%)
Collections	union	0.2	221.187	729998 (7.0%)
Collections	union	0.5	226.687	729998 (7.0%)
Collections	union	0.9	229.723	1042854 (10.0%)

Table 4.3: Perplexity of selections for TED in-domain with random replacement and pivot alignment; the target of the perplexity is TED close-in-domain

the baseline (Selection1, Selection2), the minimal perplexities with union limitation in the Papers selction are higher by about 7. However, using intersection of vocabularies yields marginaly better perplexity compared with the baseline (up to 3). Minimal perplexities of selections from Collections are higher than those of Papers. For intersection vocabulary, the difference is around seven. For unified vocabulary the difference is from 61 to 70. The size of respective selections differs between minimal perplexities of Papers and Collections.

For extrinsic evaluation, selections with minimal perplexity on the intersected vocabulary are considered. Selections with minimal perplexity on the unified vocabulary are larger and may introduce more noise. In the SMT evaluation system, configurations with the respectively trained language models and additional language model from Baseline1 are tested. Both, Papers and Collections show similar tendencies. The higher the distribution

	distribution	BLEU with BL1	BLEU without BL1
Papers	0.2	19.65 %	16.87 %
Papers	0.5	19.53 %	16.70 %
Papers	0.9	19.35 %	16.48 %
Collections	0.2	19.98 %	19.18 %
Collections	0.5	19.50 %	18.98 %
Collections	0.9	19.49 %	18.83 %

Table 4.4: Extrinsic evaluation of random replacement and pivot alignment with additional language model based on EPPS, NC and TED rest corpora (BL1) and only with language model built on the respectively selected corpus

parameter is high, the lower is the BLEU score. BLEU scores for Collections are almost consistently lower than BLEU scores for Papers. Including the language model of Baseline1, the difference in BLEU score is small. Excluding the language model of Baseline1 from the configuration results in difference of about 2.3 BLEU points. The best baseline (Selection3) is not exceeded by any of these configurations.

4.3.2 SMT

The synonym table is used in a German-German system combined with a German language model trained on the German part of the EPPS corpus. Weights are obtained from the hand-made tuning set held out from TED. The TED in-domain set is translated and the 20 best hypotheses are considered. The hypotheses are concatenated with the TED in-domain set, any duplicates are removed and subsequently used as target corpus for selection according to Moore and Lewis. Perplexities for no vocabulary limitation and

distribution	limitation	min. perplexity	selection size
Papers	intersection	111.131	787508 (9.5%)
Papers	union	224.239	828956 (10.0%)
Collections	intersection	105.849	834283 (8.0%)
Collections	union	262.084	1042854 (10.0%)

Table 4.5: Perplexity of selections for TED in-domain with SMT decoding and pivot alignment; the target of the perplexity is TED close-in-domain

union limitation show atypical behaviour for both Papers and Collections. However, the curves of perplexity for intersection limitation appears typical, even though not perfect 6.8 6.9. The perplexities are worse compared to random replacement. Collections yields a better perplexity than Papers with intersected vocabulary. With unified vocabulary, Papers gets a better perplexity.

The selected subcorpora are subsequently used for extrinsic evaluation. Two configurations are tested for Papers and Collections. The first one uses solely the built language model. The second configuration considers also considers the language model from Baseline1. On

	BLEU with BL1	BLEU without BL1
Papers	19.34 %	16.68 %
Collections	19.81 %	18.78 %

Table 4.6: Extrinsic evaluation of SMT decoding and pivot alignment with additional language model based on EPPS, NC and TED rest corpora (BL1) and only with language model built on the respectively selected corpus

Papers, paraphrasing with SMT decoding is worse than paraphrasing with random replacement. On Collections, SMT decoding outperforms random replacement with parameters 0.5 and 0.9. Collections got better results. Without the additional language model, the difference in BLEU score is higher than with the additional language model from Baseline1.

4.4 Recursive autoencoder approach

In the process of building a synonym lexicon, following thresholds are applied on metrices:

- Levenshtein ration: ≥ 0.7
- conditional probability: ≥ 0.015
- word coverage ration: ≥ 0.3
- similarity distance: ≤ 0.1

A lexicon with about 1.5M entries comes out. The nature of this lexicon is not so much synonyms but more semantic relations like name of countries or phrases belonging to the same field of topic.

4.4.1 Random replacement

Random replacement is performed with 0.2, 0.5 and 0.9 as Bernoulli-distribution parameter and 10 iterations. As before, the outcome is concatenated with TED in-domain and any duplicates are removed. The behaviour of perplexities on selection size is almost identical for all three parameters for Papers. Intersected vocabulary results in typical line of perplexity described by Moore and Lewis and no vocabulary limitation or unified vocabulary show atypical behaviour 6.10. For Collections, minimal perplexities with intersected vocabulary are about the same, whilst minimal perplexities on unified vocabulary ranges from 221 to 230. Only the patterns of perplexity for unified and intersected vocabulary result in a way described by Moore and Lewis 6.11. For all combinations, the minimal

	limitation	distr. parameter	min. perplexity	selection size
Papers	intersection	0.2	100.598	538821 (6.5%)
Papers	intersection	0.5	100.422	455926 (5.5%)
Papers	intersection	0.9	101.512	538821 (6.5%)
Papers	union	0.2	190.475	828956 (10.0%)
Papers	union	0.5	188.855	828956 (10.0%)
Papers	union	0.9	191.12	828956 (10.0%)
Collections	intersection	0.2	98.7232	417142 (4.0%)
Collections	intersection	0.5	99.1992	469284 (4.5%)
Collections	intersection	0.9	97.2917	521427 (5.0%)
Collections	union	0.2	221.187	729998 (7.0%)
Collections	union	0.5	226.687	729998 (7.0%)
Collections	union	0.9	229.723	1042854 (10.0%)

Table 4.7: Minimal perplexity of selections for TED in-domain with SMT decoding and RAE; the target of the perplexity is TED close-in-domain

perplexities are higher than minimal perplexities in the selection baselines. The minimal perplexities for Collections are consistently closer to its respective selection baselines with intersected vocabulary. With unified vocabulary, the minimal perplexities for Papers are consitently farther away from its respective selection baselines.

Selections according to minimal perplexity on intersected vocabulary are considered in the extrinsic evaluation system. Configurations with and without additional language model from Baseline1 are tested. With additional language model, BLEU scores for Collections

	distribution	BLEU with BL1	BLEU without BL1
Papers	0.2	19.44 %	16.55 %
Papers	0.5	19.39 %	16.52 %
Papers	0.9	19.49 %	16.58 %
Collections	0.2	19.07 %	17.95 %
Collections	0.5	19.24 %	16.52 %
Collections	0.9	19.26 %	17.95 %

Table 4.8: Extrinsic evaluation of random replacement and RAE with additional language model based on EPPS, NC and TED rest corpora (BL1) and only with language model built on the respectively selected corpus

are lower than BLEU scores for Papers. Without additional language model, the perplexities for Collections are higher compared to BLEU scores for Papers.

4.4.2 SMT

The German-German system, that is applied for pivot alignment, is used as well with the lexicon taken from RAE as translation model. TED in-domain is translated and the 20 best hypotheses are taken. After concatenation with TED in-domain, duplicates are removed.

For selection on Papers, unified vocabulary and no vocabulary limitation results in an atypical behavour of perplexities. Putting no limitation on the vocabulary or limiting to unified vocabulary results in an atypical behaviour of perplexities. The most restrictive limitation, intersected vocabulary, delivers a curve of perplexity that resembles the description from Moore and Lewis 6.12.
For selection on Collections, only perplexities based on no vocabulary limitation show atypical behaviour. The line of perplexity for unified vocabulary somehow resembles the typical pattern of perplexities according to Moore and Lewis. Like in Papers, only intersected vocabulary gives a line of perplexity that is typical to data selection described by Moore and Lewis 6.13. Minimal perplexities are higher than respective results obtained

	limitation	min. perplexity	selection size
Papers	intersection	107.457	704612 (8.5%)
Papers	union	220.499	828956 (10.0%)
Collections	intersection	103.962	782141 (7.5%)
Collections	union	258.739	1042854 (10.0%)

Table 4.9: Minimal perplexity of selections for TED in-domain with SMT decoding and RAE; the target of the perplexity is TED close-in-domain

with random replacement. Likewise for random replacement, minimal perplexity with intersected vocabulary is lower for Collections than for Papers but minimal perplexity with unified vocabulary is higher for Collections than for Papers. Minimal perplexities of selections from Collections get closer to its selection baselines than minimal perplexities of Papers do.

The selection with minimal perplexity on intersected vocabulary is tested in the SMT evaluation system. Configurations with and without additional language model from Baseline1 are tried. Selections on Collections show better BLEU scores than selections on Papers,

	BLEU with BL1	BLEU without BL1
Papers	19.47 %	16.62 %
Collections	19.74 %	18.59 %

Table 4.10: Extrinsic evaluation of SMT decoding and RAE with additional language model based on EPPS, NC and TED rest corpora (BL1) and only with language model built on the respectively selected corpus

in particular without additional language model. With and without additional language model and on Papers and Collections, BLEU scores outperform or lay on the upper range compared to applying random replacement.

5. Outlook

Even though improvement has been seen only in one case, some insights can be taken out of it. Selection by Moore and Lewis works worse on a noisy corpus than keeping the noisy corpus out at all. Since the selection operates on sentence level, noise within sentences will be carried on or the whole sentence is filtered out. Considering this sensitivity of selection by Moore and Lewis, noisy data must be preprocessed in a way that noise on phrase level is diminished as well. With a noise-poor corpus, selection by Moore and Lewis succeds with 1.3 BLEU points over the baseline configuration which uses the whole noise-poor corpus without any selection.
The intrinsic evaluation (perplexity) appears to be influenced by noise in a negative way. Even though the noisy corpus Papers scores lower BLEU points, the perplexities of its selections are lower compared to the corpus Collections. More noise in the data may lead to a smaller selection size for minimal perplexity. With less data lower perplexity can be gained. Regarding vocabulary limitations, intersection gives the most reliable output comparing it with the typical perplexity curve described by Moore and Lewis. So it appears that the most restrictive limitation increases robustness. This assumption goes in hand with the previous observance that selection by Moore and Lewis is not robust to noise.
The two paraphrasing techniques turn out to require a lot of improvement to produce feasable paraphrases. Random replacement shows promising advances. For pivot-alignment based lexicon, BLEU scores drop with increasing Bernoulli parameter on both corpora. For RAE based lexicon, BLEU scores inrease with increasing Bernoulli parameter on Collections corpus. On Papers corpus, no clear tendency is visible. Despite its simplicity, random replacement outperforms SMT decoding on intrinsic evaluation. On extrinsic evaluation, both techniques deliver comparable figures. SMT decoding appears to introduce less new words in the paraphrases. It also lays focus more on reordering phrases which gives a different n-gram distribution and hence a higher perplexity. In the extrinsic evaluation, this disadvantage disappears mostly.
Regarding only scores with the synonym lexicon based on pivot alignment, small improvement can be reached for the noisy Papers corpus in extrinsic and intrinsic evaluation. For the noise-poor Collections corpus, no improvement can be seen, both in the intrinsic and in the extrinsic evaluation. Also, random replacement works better than SMT decoding. With the lexicon based RAE, no improvement can be reached.
To draw a conclusion, perplexity based evaluation does not go entirely hand in hand with BLEU scoring. Even though the motivating idea sounds plausible, the realisation requires more careful design to avoid production of noise in all steps. First, paraphrasing turns out to be difficult. Not only should be the output correct in terms of syntax but also in

terms of semantics. Applying tools from SMT requires parallel data, which is not broadly available, and the consideration, that translation is performed within the same language. Specific filtering of the lexicon is necessary to reduce noise. But apparently, it is not enough.
Second, lexicon based on pivot alignment gives good synonyms. Postprocessing as displayed is nevertheless required to minimise noise. RAE based lexica do not only contain synonyms but also pairs of other semantic relations. Here, more analysis has to be performed from position and neighbours to find actual synonyms.
Third, perplexity only measures similarity based on syntax. With an improved paraphrasing technique, it is interesting to investigate if perplexity as selection criterium then correlates higher with the resulting BLEU score. In particular, exploring ways to apply perplexity effectively in noisy data might be helpful as well, given the cost in time and non optimal thresholds.

A first step towards improvement is starting at the source, that is the text extraction from PDF files. More sophisticated methods can be applied to identify actual text from mathematical equations, titles, foot notes, annotations and other formating elements. Raddum Berg et al used a Support Vector Machine (SVM) on the geometrical properties. It appears to be quite interesting to train language models on titles, text continuation and identification of mathematical symbols as well on annotations. Combined with spatical features and considering as well the preceding and succeding page should give a stronger method to filter out noise. In particular, punctuations and mathematical equations are used which often transforms to noise ("a = al 2 o 3 , s = sio 2 + p 2 o 5 , c = [...] die variation der chemischen zusammensetzung ist ...") It is nevertheless remarkable that with the approach of Moore and Lewis noisy sentences can be filtered out. Decreasing the noise in the text extraction of PDF files does not only aim on having less noise in the data because this can be removed by applying Moore and Lewis' method. But it primarily should deliver more meaningful text from a resource which is rather domain specific, limited and therefore more expensive.
Another step is to refine the methods for harvesting or producing paraphrases. Because the level of granularity is pretty deep, models must be sensitive to operate as exact as possible. Decoding systems described in this thesis only rely on language and translation models. Additional models are not considered. Reordering models, any post-processing, labeling or factorised models might help to produce better paraphrasing. This will ultimatively help to raise the quality of in-domain corpus using the approach of Moore and Lewis.
The recursive autoencoder architecture can be improved as well. The idea of multitasking from Collobert and Weston can be applied on recursive autoencoders too. The difficulty is to allow variable input length for tasks which require as well context information. Giving more information at hand may push semantic representation even further. Other neural network architectures like recurrent neural networks handle variable input length as well and have achieved good results as shown by Irsoy and Cardie[15]. So far, the proposed models deduce from singular instances of phrases without considering the context of these particular phrases. With words having only one semantic meaning, this concept will work properly. But since sensitivity is important for semantic analysis on word level, context becomes more important to be aware of. To acomplish it, Paulus et al suggest a new recursive neural network architecture [26]. Rather using a strictly feed forwarding architecture, the inference of semantic representations involves considering representations of topology-wise higher terms, that is feedbackwarding is performed. Paulus et al report an improvement over the feedforward recursive neural networks. Although their objective is sentiment prediction and not direct semantic representations.
At last, the process of selection can be modified. Mediani et al report an increase in performance when decreasing the size of the domain-specific corpus. It comes also closer to many scenarios where not much data for the translation system is at hand. Improve-

ment may not only be reached by a smaller domain-specific corpus but also through closer relationship between domain-specific and general corpus. In particular, Papers might be too distant to TED in-domain in terms of semantics. Testing the presented techniques for two corpora more close to each other, like Papers as general corpus and some lecture transcripts as domain-specific corpus, might result in better outcome.

6. Appendix

6.1 Recursive Autoencoder

Recursive neural networks are a variation of feed forward neural networks. Rather applying different layers subsequently on the respective output, only one layer is used to merge two output instances together to eventually obtain one output instance. The order of merging goes according to a specific hierarchy, henceforth called *topology instance.*
The topology instance can be either derived from an heuristic algorithm or obtained from a grammatical parser. A recursive neural network can be seen as well as an autoencoder with dynamically dimensioned input. In the fashion of autencoder, training uses reconstruction as objective. Hence an additional layer of weights is required. Finding the semantic representation of two phrases corresponds to encoding and decomposing a phrase into two semantic entities is adequate to decoding. Socher et al [28] propose such a neural network architecture to measure the semantic content of phrases. Although Socher et al offer code for performing encoding with a given model of parameters, no code for training is published (effective May 1st 2015). Therefore, an own implementation is presented for both model training and semantic analysis.

6.1.1 Architecture

Like the name tells it, the neural network is applied recursively, folding a tree of semantic values. Values at its leafs are the semantic representation of the corresponding words or phrases. Merging two nodes in the tree results in a semantic representation for the phrase which is spanned over the words at the leafs of the subtrees both merging nodes are root of. The neural network, therefore, can be described as a function θ which maps two instances of a semantic space into the very same space. The initial semantic representations for words are taken from a lookup table which holds the semantic representations for a closed vocabulary set. Unknown words are assigned to the null vector. That is no activation is triggered for unknown words. Google's word-embedding tool *word2vec* offers a way to compute this particular lookup table in an easy way.

The choosen tree topology is binary. The reason therefore is almost only simplicity. An architecture with more than two child nodes could be used as well. Grammatical parsers themselves produce syntax trees with varying amount of child nodes. Some architecture which is able to consider such kinds of syntax tree should - in theory - delivers better results. The complexity herein lies in merging an arbitrary amount of child nodes such that weights reflect and organise themselves to cope with this inconsistency. Expecting

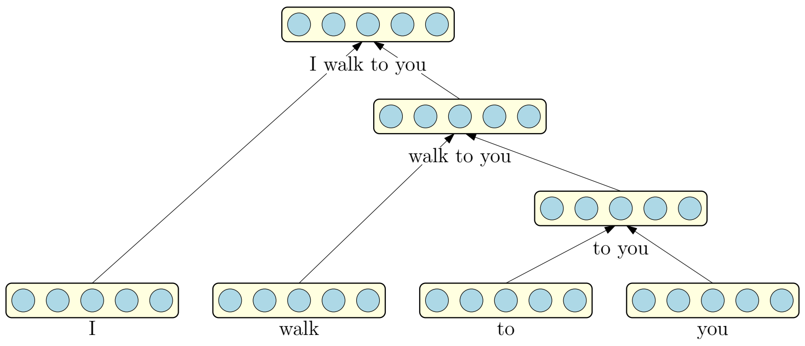

Figure 6.1: A recursive autoencoder takes a sentence of arbitrary length and folds it according to some tree topology. The nodes' values are semantic representations of the respective phrases. Semantic representations for input words can be

a fixed amount of child nodes brings along the issue that grammar of natural languages cannot be correctly represented by trees with fixed amount of child nodes per parent node. Hence, architectures of such kind need routines how to solve situations where the actual amount of child nodes differs from the branching factor of the merging tree. If more child nodes in the syntax tree exist than merging is expecting, these child nodes have to be regrouped in an hierarchy which will introduce noise since the hierarchy is artifical. Additionally, the last merge in the made up hierarchy might have less nodes than expected. This also happens if fewer child nodes exist in the syntax tree than merging is expecting. Using a branching factor of two, that is using a binary tree, will solve latter problem. Merging a specific amount of nodes in binary mode works that two nodes are merged and replaced by the merging result. Hence, no node will be left. The artifical hierarchy is either simple, that is starting from one end going to the other one, or more complex, that is employing some heuristics to find the least noise producing hierarchy. Heuristics commonly depend on the model which in training is about to be formed. It can cause complexity and unnecessary noise. In our approach, we omit heuristics for these reasons and apply a left to right merging process.

6.1.1.1 Untied input layer dimension

A variation of the standard recursive neural network employs a different input dimension on the first layer. Rather mapping into the same space, the semantic value representations reside in a differently dimensioned space. It can be both higher or lower. Consequently, semantic representations at the leaf nodes cannot be simply compared with semantic representations at the branch nodes due to their different dimensions. Lowering dimension can find use in making representations on higher level more compact. More low frequent noise will be filtered out and computation time will decrease. In combination with an output function at all or some nodes, a reduced inner semantic space can bring improvement. Socher et al perform logical regression on a recursive autoencoder for sentiment prediction[29]. On the last merged node, a softmax layer is placed to carry out predictions for different sentiments. The training objective is to minimise the overall prediction error as well as the reconstruction error. The output does not directly uses the semantic representations. So, the dimension of the inner semantic space is free to vary.

Using higher dimensions as well can make sense. For reasons of grasping more information, the dimension of the inner semantic space can be increased. Mechanisms have to be put in place to avoid overfitting. The result is a sparse autoencoder. Via additional penalty constraints in the objective function, the weights are encouraged to have an impact which is as little as possible.

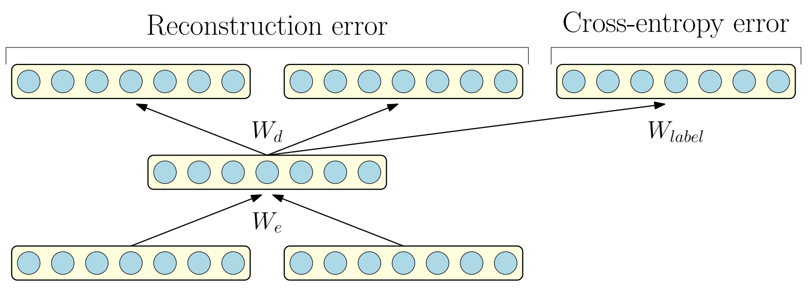

Figure 6.2: A recursive autoencoder can be extended with labeling, that is classification, for all or only the root node. The training objective then is the labeling error plus the reconstruction error. Socher et al use classification of semantic representations for predicting sentiment distributions.

Low dimensioned semantic spaces aim towards removing low frequencies in the data. It goes according with the idea that the general structure is hidden by low frequencies, like typos or unusual expressions. The more frequent some data appears, the more likely is it to be important and to be not noise. Any correct data which seldom appears will be removed like it was noise. With highly dimensioned inner space, the intention is to capture small details much better. The issue, of course, is to decide what is likely to be noise and what is real data. Defining the separation between data and noise makes the design of sparse autoencoders difficult.

Since our target is to compare phrases semantically, semantic representations have to be comparable as well. We also do not want to restrict ourself to compare only merged phrases but also intent to include word representations in the comparisons. Hence, recursive autoencoder we deploy will have a tied input layer dimension and will use only one set of parameters. For the sake of illustrating recursive neural networks, untied variations have been discussed.

6.1.1.2 Reconstruction

Training goes according to some objective function which is aimed to get minimised. In a standard recursive autencoder with no additional layers on top, the training objective J_{SRAE} is the direct reconstruction error; that is the sum over the euclidean distances between the actual and the decoded child nodes of each parent node $\tau \in \mathcal{P}$ in the folding tree.

$$J_{\text{SRAE}} = \sum_{\tau \in \mathcal{P}} \|x^{(l(\tau))} - \hat{x}^{(l(\tau))}\|_2^2 + \|x^{(r(\tau))} - \hat{x}^{(r(\tau))}\|_2^2 \tag{6.1}$$

Two problems arise. The first problem is that reconstructing only the direct child nodes means equal contribution of the respective reconstruction errors disregarding the size of the respective subtrees. If the first child node represents one word and the second child node represents a phrase of ten words, reconstructing solely these representators will make paying the same attention towards the word of the first node as towards the ten words of the second node. This misproportion leads to better encoding for words close to the root node whilst words far away from the root node. Consequently, the semantic representation of the whole sentence is dominated by words close to the root node.
The second problem is caused by the fact that training computes the parent nodes which it then tries to reconstruct. The training algorithm can achieve a lower reconstruction error by lowering the norms of the branching nodes, that is weights tend towards zero which is also called weight implosion. To overcome this issue, Socher et al[28] suggest adding a normalisation layer on top of each parent node which will ensure that any representation

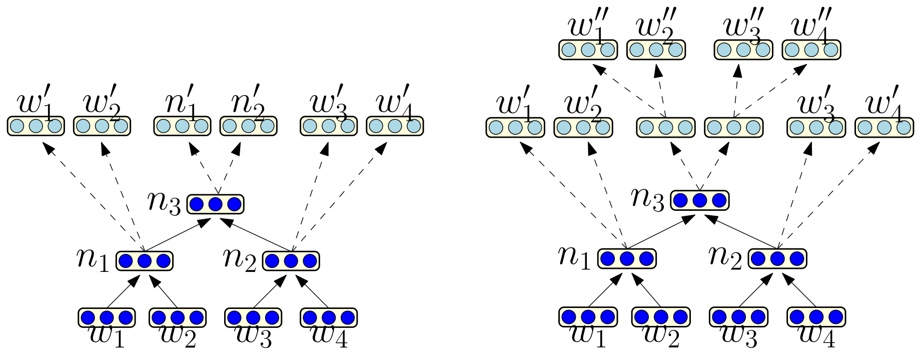
w'_1 w'_2 n'_1 n'_2 w'_3 w'_4
n_3
n_1 n_2
w_1 w_2 w_3 w_4
w''_1 w''_2 w''_3 w''_4
w'_1 w'_2 w'_3 w'_4
n_3
n_1 n_2
w_1 w_2 w_3 w_4

Figure 6.3: Standard recursive autoencoder decodes only the direct children for all its branch nodes. Unfolding recursive autoencoder decodes fully to the leaf nodes for all its branch nodes. The reconstruction error sums over all leaf nodes of the decoding trees.

has exactly length one and does not diminish towards zero.

An alternative to employing direct reconstruction error is to fully unfold the subtree of each parent node $\tau \in \mathcal{P}$ and then to sum over the reconstruction errors at the leaf nodes $\pi \in \mathcal{L}(\tau)$ of each subtree with root τ.

$$J_{\text{URAE}} = \sum_{\tau \in \mathcal{P}} \sum_{\pi \in \mathcal{L}(\tau)} \|x^{(\pi)} - \hat{x}^{(\pi)}\|_2^2 \tag{6.2}$$

Because the training algorithm tries to reconstruct in unfolding mode the leafs which of course are fixed, exploding or imploding weights do not bring any advantages in the task to minimise the training objective J_{URAE}. Thus, no normalisation layer has to be put on the parent nodes. The other problem of the standard recursive autoencoder has been faced as well. The size of the reconstruction error depends on the magnitude of the individual reconstruction errors as well on the amount of leaf nodes to be reconstructed. Merging two nodes of which one represents one word and the other node represents ten words, the reconstruction error of the latter will be in general higher than the reconstruction error of the first one. The training algorithm will therefore pay more attention towards the node with ten words. Weights are also trained such that the model does not overwhelmingly considers words close to the root, but allows for words farther away from the root to contribute substantially to the eventual semantic representation of the sentence.

6.1.1.3 Deep network

The recursive autoencoder as presented so far belongs to the class of shadow networks. No actual latent or hidden layers are used in this particular architecture. Similar to other shallow network architectures, it can be extended to a deep network architecture which holds one or more hidden layers. The idea of such hidden layers is to find more abstract features of the input whilst traversing through the layers of the network. Deep networking can also be seen as a technique to pass through features of features which actually is equivalent to finding more abstract features for describing the input. Each layer in a deep recursive autoencoder holds representations according to the topology of the merging tree. Thereby, a node τ in layer i not only depends on its child nodes $l(\tau)$ and $r(\tau)$ in the same layer i but also on the equivalent node τ in the lower layer $i-1$.

$$x^{(\tau,i)} = f(L^{(i)}x^{(l(\tau),i)} + R^{(i)}x^{(r(\tau),i)} + P^{(i-1)}x^{(\tau,i-1)} + b^{(i)}) \tag{6.3}$$

The lowest level $i = 0$ has no lower level by nature. So it gets simplified to the dependencies like in the shallow network architecture.

$$x^{(\tau,0)} = f(L^{(0)}x^{(l(\tau),0)} + R^{(0)}x^{(r(\tau),0)} + b^{(0)}) \tag{6.4}$$

Each layer operates in its own semantic space and can have its dimensionality. To filter more and more information out traversing through the layers, the dimensions are designed to decrease going up the the stack of layers. Weights $P^{(i)}$ map representations from the semantic space of the lower layer into the semantic space of the upper layer. Representations of the subphrases are taken from the uppermost layer.
Training is performed layerwise. That is first, weights of the lowest layer are trained with the objective to minimise the reconstruction error solely on this layer. After enough iterations, any input is put through the lowest layer and its results are present to the second lowest layer whose weights are trained subsequently with the objective to minimise the reconstruction error on that layer. From the second lowest layer on, leaf nodes are all nodes in the previous layer. Reconstruction error then is computed based on this set of nodes rather nodes in the current layer representing single words.
Despite successful results of deep network architectures in other domains, Socher et al[28] report that they see not any improvement using deep recursive autoencoders. It appears rather that due to complexity, the deep recursive autoencoder gets stucked more easily in local minima.

6.1.2 Mathematical background

A shallow recursive autoencoder encodes a parent node τ with its left child node $l(\tau)$ weighted by L, its right child node $r(\tau)$ weighted by R and some bias b whose result is transformed by the activation function f.

$$x^{(\tau)} = f(Lx^{(l(\tau))} + Rx^{(r(\tau))} + b) \tag{6.5}$$

In the standard version, decoding is limited to only the direct children. Therefore, decoded child nodes $l(\tau)$ and $r(\tau)$ solely depend on the parent node τ which contributes with respective weights $\hat{L}$ and $\hat{R}$ as well on biases $c^{(l)}$ and $c^{(r)}$ respectively.

$$\hat{x}^{(l(\tau))} = f(\hat{L}x^{(\tau)} + c^{(l)}) \tag{6.6}$$

$$\hat{x}^{(r(\tau))} = f(\hat{R}x^{(\tau)} + c^{(r)}) \tag{6.7}$$

The objective function of training a standard recursive autoencoder is the overall sum of euclidean distances between the actual and reconstructed child nodes of each parentnode. For the purpose of easier differentitation, factor $\frac{1}{2}$ is added.

$$J_{\text{SRAE}} = \frac{1}{2} \sum_{\tau \in \mathcal{P}} \|x^{(l(\tau))} - \hat{x}^{(l(\tau))}\|^2 + \|x^{(r(\tau))} - \hat{x}^{(r(\tau))}\|^2 \tag{6.8}$$

Applying backpropagation in model training requires to find out how much each weight contributes to the overall reconstruction error J_{SRAE}. Differentiating the objective function to the decoding bias, that is $c^{(l)}$ or $c^{(r)}$, only takes to traverse over all parent nodes, measuring the respective distance of actual and decoded child nodes and scaling it up entrywise by the ascend of the activation function f.

$$\nabla_{c^{(l)}} J_{\text{SRAE}} = - \sum_{\tau \in \mathcal{P}} \left[(x^{(l(\tau))} - \hat{x}^{(l(\tau))}) \otimes f'(\hat{L}x^{(\tau)} + c^{(l)})\right]^T \tag{6.9}$$

$$\nabla_{c^{(r)}} J_{\text{SRAE}} = - \sum_{\tau \in \mathcal{P}} \left[(x^{(r(\tau))} - \hat{x}^{(r(\tau))}) \otimes f'(\hat{R}x^{(\tau)} + c^{(r)})\right]^T \tag{6.10}$$

Gradient for the decoding weights $\hat{L}$ and $\hat{R}$ works almost similar. For each left node, the reconstruction error is scaled entrywise by the derivation of the activation function and with the left node representation, the outer product is built. Summing over all left nodes results in the gradient of the left decoding weights. Computing the gradient for the right decoding weights works analogously.

$$D_{\hat{L}} J_{\text{SRAE}} = -\sum_{\tau \in \mathcal{P}} \left[(x^{(l(\tau))} - \hat{x}^{(l(\tau))}) \otimes f'(\hat{L}x^{(\tau)} + c^{(l)}) \right] \left[x^{(l(\tau))} \right]^T \tag{6.11}$$

$$D_{\hat{R}} J_{\text{SRAE}} = -\sum_{\tau \in \mathcal{P}} \left[(x^{(r(\tau))} - \hat{x}^{(r(\tau))}) \otimes f'(\hat{R}x^{(\tau)} + c^{(r)}) \right] \left[x^{(r(\tau))} \right]^T \tag{6.12}$$

Involving length normalisation in the encoding makes the case rather more complicated, although it follows the same pattern as without length normalisation. For better illustration, the gradient of encoding bias and weights without length normalisation is derived. First, some auxiliary variables are defined. The activation $a^{(\tau)}$ of a node is the raw representation value before feeding it entrywise to the activation function f.

$$a^{(\tau)} = Lx^{(l(\tau))} + Rx^{(r(\tau))} + b \tag{6.13}$$

If length normalisation is emploid, the direct output of the activation function is labeled $y^{(\tau)}$ and the actual representation value $x^{(\tau)}$ then will not only have dependency on same components of $y^{(\tau)}$ but will need all components of $y^{(\tau)}$ to compute at least one of its components.

$$y^{(\tau)} = f(a^{(\tau)}) \tag{6.14}$$

$$x^{(\tau)} = \frac{y^{(\tau)}}{\|y^{(\tau)}\|} \tag{6.15}$$

First, the gradient of the encoding bias is derived from the objective function J_{SRAE} without employing length normalisation. The Jacobian matrix $D_b x^{(\tau)}$ of some encoding node representation $x^{(\tau)}$ regarding the encoding bias, builds a simple linear combination of the Jacobian matrices $D_b x^{(l(\tau))}$ and $D_b x(r(\tau))$ of its left and right child nodes.

$$D_b x^{(\tau)} = \text{diag}(f'(a^{(\tau)})) \cdot L \cdot D_b x^{(l(\tau))} + \text{diag}(f'(a^{(\tau)})) \cdot R \cdot D_b x^{(r(\tau))} + \text{diag}(f'(a^{(\tau)})) \tag{6.16}$$

The Jacobian matrix $D_b x^{(\tau)}$ of any leaf node τ will be zero and therefore will terminate the recursion. The gradient $\nabla_b J_{\text{SRAE}}(\tau)$ for reconstructing only one node τ and differentiating it to the encoding bias b is as well a linear combination of the Jacobian matrizes $D_b x^{(l(\tau))}$ and $D_b x^{(r(\tau))}$ of its left and right child nodes.

$$\begin{aligned} \nabla_b J_{\text{SRAE}}(\tau) &= z_l(\tau) D_b x^{(l(\tau))} + z_r(\tau) D_b x^{(r(\tau))} - \beta(\tau)^T \text{diag}(f'(a^{(\tau)})) \\ \text{with} \quad & \\ z_l(\tau) &= [\varepsilon_l(\tau) - \beta(\tau)\ \text{diag}(f'(a^{(\tau)}))L]^T \\ z_l(\tau) &= [\varepsilon_r(\tau) - \beta(\tau)\ \text{diag}(f'(a^{(\tau)}))R]^T \\ \beta(\tau) &= \varepsilon_l(\tau)\ \text{diag}(f'(\hat{a}^{(l(\tau))}))\hat{L} + \varepsilon_r(\tau)\ \text{diag}(f'(\hat{a}^{((r(\tau))}))\hat{R} \\ \varepsilon_l(\tau) &= x^{(l(\tau))} - \hat{x}^{(l(\tau))} \\ \varepsilon_r(\tau) &= x^{(r(\tau))} - \hat{x}^{(r(\tau))} \end{aligned} \tag{6.17}$$

For the overall gradient $\nabla_b J_{\text{SRAE}}$ of reconstructing all parent nodes in a tree, all gradients of reconstructing a single node are accumulated.

$$\nabla_b J_{\text{SRAE}} = \sum_{\tau \in \mathcal{P}} \nabla_b J_{\text{SRAE}}(\tau) \tag{6.18}$$

The linear combination of the recursive gradients can be exploited to obtain a recursive function ξ_b which starts at the root node and propagates towards the leafs while leaving behind summands whose accumulated result in the wanted gradient.

$$\xi_b(\tau,\mu) = \begin{cases} 0, & \text{if } \tau \text{ is leaf} \\ \xi_b(l(\tau), p_l(\tau,\mu)) + \xi_b(r(\tau), p_r(\tau,\mu)) + p_o(\tau,\mu), & \text{else} \end{cases}$$

$$\text{with} \qquad \begin{aligned} p_l(\tau,\mu) &= \varepsilon_l(\tau)^T + (\mu - \beta(\tau)^T)\ \mathrm{diag}(f'(a^{(\tau)}))L \\ p_r(\tau,\mu) &= \varepsilon_r(\tau)^T + (\mu - \beta(\tau)^T)\ \mathrm{diag}(f'(a^{(\tau)}))R \\ p_o(\tau,\mu) &= (\mu - \beta(\tau)^T)\ \mathrm{diag}(f'(a^{(\tau)})) \end{aligned} \tag{6.19}$$

Mathematical induction helps to conclude that for each node π and some factor μ, function $\xi(\pi,\mu)$ holds the sum over all reconstruction gradients $\nabla_b J_{\mathrm{SRAE}}(\tau)$ of parent nodes τ in the subtree rooted by π and the Jacobian matrix $D_b x^{(\pi)}$ weighted by μ.

$$\xi(\pi,\mu) = \sum_{\tau\in\mathcal{P}(\pi)} \nabla_b J_{\mathrm{SRAE}}(\tau) + \mu D_b x^{(\pi)} \tag{6.20}$$

The basis case is a simple tree with just one branch node π which refers to two child nodes. Due to both child nodes being leafs, the recursion does not exist and the term can be reduced easily.

$$\begin{aligned} \xi_b(\pi,\mu) &= \xi_b(l(\pi), p_l(\pi,\mu)) + \xi_b(r(\pi), p_r(\pi,\mu)) + p_o(\pi,\mu) \\ &= (\mu - \beta(\pi)^T)\ \mathrm{diag}(f'(a^{(\pi)})) \end{aligned} \tag{6.21}$$

$$\begin{aligned} \sum_{\tau\in\mathcal{P}(\pi)} \nabla_b J(\pi) + \mu D_b x^{(\pi)} &= \nabla_b J(\pi) + \mu D_b x^{(\pi)} \\ &= -\beta(\pi)^T\ \mathrm{diag}(f'(a^{(\pi)})) + \mu\ \mathrm{diag}(f'(a^{(\pi)})) \\ &= (\mu - \beta(\pi)^T)\ \mathrm{diag}(f'(a^{(\pi)})) \end{aligned} \tag{6.22}$$

For two arbitrary trees with respective roots π_1 and π_2 and some arbitrary factor μ, it is assumed that both hold the hypothesised equation individually.

$$\xi_b(\pi_i,\mu) = \sum_{\tau\in\mathcal{P}(\pi_i)} \nabla_b J(\tau) + \mu D_b x^{(\pi_i)}, \text{ for } i = 1,2 \tag{6.23}$$

The induction step then is taken, by putting both trees together with a new root node π having π_1 to its left and π_2 to its right. Factor μ is supposed to be arbitrary. Decomposing into the linear combination gives the opportunity to apply the assumption which eventually produces the overall sum of partial gradients $\nabla_b J(\tau)$ and the Jacobian matrix $D_b x^{(\pi)}$ weighted by μ.

$$\begin{aligned} \xi_b(\pi,\mu) &= \xi_b(\pi_1, p_l(\pi,\mu)) + \xi_b(\pi_2, p_r(\pi,\mu)) + p_o(\pi,\mu) \\ &= \sum_{\tau\in\mathcal{P}(\pi_1)} \nabla_b J(\tau) + p_l(\pi,\mu) D_b x^{(\pi_1)} + \sum_{\tau\in\mathcal{P}(\pi_2)} \nabla_b J(\tau) + p_r(\pi,\mu) D_b x^{(\pi_2)} + p_o(\pi,\mu) \\ &= \sum_{\tau\in\mathcal{P}(\pi)} \nabla_b J(\tau) - \nabla_b J(\pi) + \mu D_b x^{(\pi)} + \nabla_b J(\pi) \\ &= \sum_{\tau\in\mathcal{P}(\pi)} \nabla_b J(\tau) + \mu D_b x^{(\pi)} \end{aligned} \tag{6.24}$$

Using $\mu = \vec{0}$ as factor for the root node π from where as well the recursion starts, delivers the overall sum of gradients for partial reconstruction.

$$\xi_b(\pi, \vec{0}) = \sum_{\tau \in \mathcal{P}} \nabla_b J(\tau) \tag{6.25}$$

For left and right encoding weights, L and R, deriving gradients works similar to deriving the gradient for encoding bias. Only the gradient $D_L J_{\text{SRAE}}$ for the left encoding weights is derived. More details on deriving the right encoding weights can be studied in the appendix since it follows the same scheme. The gradient $D_L J_{\text{SRAE}}$ for left encoding weights is the overall sum of gradients $D_L J_{\text{SRAE}}(\tau)$ of reconstructing only parent nodes τ.

$$D_L J_{\text{SRAE}} = \sum_{\tau \in \mathcal{P}} D_L J_{\text{SRAE}}(\tau) \tag{6.26}$$

The gradient $D_L J_{\text{SRAE}}(\tau)$ for reconstructing node τ is a linear combination of the partial differentiation of child node representations for left encoding weights.

$$\begin{aligned}
D_L J_{\text{SRAE}}(\tau) = & \left[\textstyle\sum_k h_k^{(l)}(\tau) \cdot \frac{\partial x_k^{(l(\tau))}}{\partial L_{ij}} + h_k^{(r)}(\tau) \cdot \frac{\partial x_k^{(r(\tau))}}{\partial L_{ij}}\right]_{ij} + H^{(o)}(\tau) \\
& \text{with} \\
h^{(l)}(\tau) = & \ \varepsilon_l(\tau) + \left[\left(\hat{R} \cdot \text{diag}(f'(a^{(\tau)})) \cdot L\right)^T \left(\varepsilon_r(\tau) \odot f'(\hat{a}^{(r(\tau))})\right)\right] + \\
& \left[\left(\hat{L} \cdot \text{diag}(f'(a^{(\tau)})) \cdot L\right)^T \left(\varepsilon_l(\tau) \odot f'(\hat{a}^{(l(\tau))})\right)\right] \\
h^{(l)}(\tau) = & \ \varepsilon_r(\tau) + \left[\left(\hat{R} \cdot \text{diag}(f'(a^{(\tau)})) \cdot R\right)^T \left(\varepsilon_r(\tau) \odot f'(\hat{a}^{(r(\tau))})\right)\right] + \\
& \left[\left(\hat{L} \cdot \text{diag}(f'(a^{(\tau)})) \cdot R\right)^T \left(\varepsilon_l(\tau) \odot f'(\hat{a}^{(l(\tau))})\right)\right] \\
H^{(o)}(\tau) = & \left[\left(\hat{L}^T \left(\varepsilon_l(\tau) \odot f'(\hat{a}^{(l(\tau))})\right)\right) \odot f'(a^{(\tau)})\right] \otimes x^{(l(\tau))} + \\
& \left[\left(\hat{R}^T \left(\varepsilon_r(\tau) \odot f'(\hat{a}^{(r(\tau))})\right)\right) \odot f'(a^{(\tau)})\right] \otimes x^{(r(\tau))}
\end{aligned} \tag{6.27}$$

The partial differentation $\frac{\partial x_k^{(\tau)}}{\partial L_{ij}}$ of child node's τ representation for left encoding weight L_{ij} itself is a linear combination of the partial differentation of its child nodes' representations.

$$\frac{\partial x_k^{(\tau)}}{\partial L_{ij}} = f'(a_k^{(\tau)}) \left[\sum_l L_{kl} \cdot \frac{\partial x_l^{(l(\tau))}}{\partial L_{ij}} + R_{kl} \cdot \frac{\partial x_l^{(r(\tau))}}{\partial L_{ij}}\right] + f'(a_i^{(\tau)}) x_j^{(l(\tau))} \tag{6.28}$$

Correspondingly to computing the gradient for encoding bias, a recursive function $\xi_L(\tau, \mu)$ can be defined which starts at node τ and goes down recursively in the subtree of τ being root. On leaf nodes, a zero matrix is returned. On branch nodes, a summand matrix is returned as well the child nodes are called with ξ and new factors.

$$\begin{aligned}
\xi_L(\tau, \mu) = & \begin{cases} 0, & \text{if } \tau \text{ is leaf} \\ \xi_L(l(\tau), q_l(\tau, \mu)) + \xi_b(r(\tau), q_r(\tau, \mu)) + q_o(\tau, \mu), & \text{else} \end{cases} \\
& \text{with} \\
& q_l(\tau, \mu) = h^{(l)}(\pi) + L^T \cdot \left(\mu \otimes f'(a^{(\pi)})\right) \\
& q_r(\tau, \mu) = h^{(r)}(\pi) + R^T \cdot \left(\mu \otimes f'(a^{(\pi)})\right) \\
& q_o(\tau, \mu) = H^{(o)}(\pi) + \left(\mu^T \cdot \vec{1}\right) \left(f'(a^{(\pi)}) \otimes x^{(l(\pi))}\right)
\end{aligned} \tag{6.29}$$

It can be proven via mathematic induction that $\xi_L(\pi,\mu)$ accumulates all partial reconstruction gradients $D_L J_{\text{SRAE}}(\tau)$ within the subtree and the partial differentation of its representation $\frac{\partial x_k^\pi}{\partial L_{ij}}$ weighted by μ.

$$\xi_L(\pi,\mu) = \sum_{\tau\in\mathcal{P}(\pi)} D_L J(\tau) + \left[\sum_k \mu_k \cdot \frac{\partial x_k^\pi}{\partial L_{ij}}\right]_{ij} \tag{6.30}$$

The proove follows the scheme as presented for the encoding bias. Details can be found in the appendix.
Applying length normalisation changes equations only slightly. The partial differentation of node representations, that is $\frac{\partial x_i^{(\tau)}}{\partial b_j}$ and $\frac{\partial x_i^{(\tau)}}{\partial L_{jk}}$, are multiplied with some denormalising matrix $D_{x^{(\tau)}} y^{(\tau)}$.

$$D_{x^{(\tau)}} y^{(\tau)} = \frac{1}{\|y^{(\tau)}\|} E - \frac{4}{\|y^{(\tau)}\|_3} \left(y^{(\tau)} \otimes y^{(\tau)}\right) \tag{6.31}$$

The second architecture's version is an unfolding recursive neural network. No length normalisation is required as discussed before. What remains, are unfolds on the tree for each node. That is, for each parent node in the tree, the complete subtree is unfolded and the distances of actual and reconstructed leaf nodes are computed to obtain the objective function J_{URAE}. As before, representations $x^{(\tau)}$ of some node τ is encoded by a linear combination of the respective child nodes $l(\tau)$ and $r(\tau)$.

$$x^{(\tau)} = f(a^{(\tau)}) = f(Lx^{(l(\tau))} + Rx^{(r(\tau))} + b) \tag{6.32}$$

Unfolding some node τ depends on the relative position to its parent node $p(\tau)$, that is it is either left or right. Both ways, the decoded node is a linear combination of its parent node as well.

$$\begin{aligned} c^{(l(\tau))} &= \hat{L}y^{(\tau)} + \hat{l} \\ c^{(r(\tau))} &= \hat{R}y^{(\tau)} + \hat{r} \\ y^{(\tau)} &= f(c^{(\tau)}) \end{aligned} \tag{6.33}$$

The representation $y^{(\pi)}$ of the root node π in a subtree is equal to the representation $x^{(\pi)}$ of the encoded node, that is $y^{(\pi)} = x^{(\pi)}$. The objective function J_{URAE} can be decomposed into the sum of objective functions $J_{\text{URAE}}(\pi)$ for parent nodes π.

$$J_{\text{URAE}}(\pi) = \sum_{\tau\in\mathcal{L}(\pi)} \frac{1}{2}\|y^{(\tau)} - x^{(\tau)}\|^2 \tag{6.34}$$

The decoding biases $\hat{l}$ and $\hat{r}$ and decoding weights $\hat{L}$ and $\hat{R}$ only depend on the subtree. Gradients of the objective function for decoding parameters start off with the error distance as weight and go up recursively in the unfolded tree.

$$\begin{aligned} \frac{\partial J_{\text{URAE}}(\pi)}{\partial \hat{r}_j} &= \sum_{\tau\in\mathcal{L}(\pi)} \sum_i \left(y_i^{(\tau)} - x_i^{(\tau)}\right) \frac{\partial y_i^{(\tau)}}{\partial \hat{r}_j} \\ \frac{\partial J_{\text{URAE}}(\pi)}{\partial \hat{l}_j} &= \sum_{\tau\in\mathcal{L}(\pi)} \sum_i \left(y_i^{(\tau)} - x_i^{(\tau)}\right) \frac{\partial y_i^{(\tau)}}{\partial \hat{l}_j} \\ \frac{\partial J_{\text{URAE}}(\pi)}{\partial \hat{L}_{jk}} &= \sum_{\tau\in\mathcal{L}(\pi)} \sum_i \left(y_i^{(\tau)} - x_i^{(\tau)}\right) \frac{\partial y_i^{(\tau)}}{\partial \hat{L}_{jk}} \\ \frac{\partial J_{\text{URAE}}(\pi)}{\partial \hat{R}_{jk}} &= \sum_{\tau\in\mathcal{L}(\pi)} \sum_i \left(y_i^{(\tau)} - x_i^{(\tau)}\right) \frac{\partial y_i^{(\tau)}}{\partial \hat{R}_{jk}} \end{aligned} \tag{6.35}$$

Even though computing the gradient of the objective function works the same for all four decoding parameters, the left decoding bias and weights are illustrated here to show the different approaches for matrix and vector differentitation. The gradients of the objective function depend on the differentiations $\frac{\partial y_i^{(\tau)}}{\partial \hat{l}_j}$ and $\frac{\partial y_i^{(\tau)}}{\partial \hat{L}_{jk}}$, respectively. The decoding subtree is folded together, that is two nodes $l(\tau)$ and $r(\tau)$ are merged together and - besides a weighted dependency on the parent node τ - leave behind a term which contributes towards the respective gradient.

$$\begin{aligned}\frac{\partial\left(y_j^{(l(\tau))}+y_i^{(r(\tau))}\right)}{\partial \hat{l}_j} &= f'(c_i^{(l(\tau))})\delta(i,j)+\sum_k\left[f'(c_i^{(l(\tau))})\hat{L}_{ik}+f'(c_i^{(r(\tau))})\hat{R}_{ik}\right]\frac{\partial y_k^{(\tau)}}{\partial \hat{l}_j}\\ \frac{\partial\left(y_j^{(l(\tau))}+y_j^{(r(\tau))}\right)}{\partial \hat{L}_{jk}} &= f'(c_i^{(l(\tau))})y_k^{(\tau)}\delta(i,j)+\sum_l\left[f'(c_i^{(l(\tau))})\hat{L}_{il}+f'(c_i^{(r(\tau))})\hat{R}_{il}\right]\frac{\partial y_l^{(\tau)}}{\partial \hat{L}_{jk}}\end{aligned} \tag{6.36}$$

The intention of the recursive merging is to retrieve an expression which only depends on the root node π of the decoding subtree because it is the only node in the subtree whose gradient itself does depend neither on any other node's gradient in the decoding subtree nor on any decoding parameters. An effective way of computing the gradients of biases and weights is to go bottom-up in the decoding tree and storing a weights vector $\lambda(\tau)$ for each node τ. The weights vectors $\lambda(\tau)$are the same for biases and weights. The computation of weights starts at the leafs of the decoding subtree with the respective reconstruction distance. Weights of some parent node τ is a linear combination of weights from the child nodes $l(\tau)$ and $r(\tau)$.

$$\lambda(\tau)=\begin{cases}(y^{(\tau)}-x^{(\tau)})^T, & \text{if } \tau \text{ is leaf}\\ \left(\lambda(l(\tau))\odot f'(c^{(l(\tau))})\right)\hat{L}+\left(\lambda(r(\tau))\odot f'(c^{(r(\tau))})\right)\hat{R}, & \text{else}\end{cases} \tag{6.37}$$

Once the subtree of weights is put together, the previously mentioned terms are added according to the weights of the respective nodes.

$$\begin{aligned}\nabla_{\hat{l}} J_{\text{URAE}}(\pi) &= \sum_{\tau\in\mathcal{L}(\pi)}(y^{(\tau)}-x^{(\tau)})^T D_{\hat{l}} y^{(\tau)} \overset{!}{=} \sum_{\tau\in\mathcal{P}(\pi)}\lambda(l(\tau))\odot f'(c^{(l(\tau))})\\ D_{\hat{L}} J_{\text{URAE}}(\pi) &= \sum_{\tau\in\mathcal{L}(\pi)}\sum_i (y_i^{(\tau)}-x_i^{(\tau)})\frac{\partial y_i^{(\tau)}}{\partial \hat{L}_{jk}}\\ &\overset{!}{=}\sum_{\tau\in\mathcal{P}(\pi)}\left(\lambda(l(\tau))\odot f'(c^{(l(\tau))})\right)\otimes y^{(\tau)}\end{aligned} \tag{6.38}$$

The equality relation can be proven via mathematical induction. The simplest case is a tree which has root π and two leafs $\tau_1 = l(\pi)$ and $\tau_2 = r(\pi)$. Writing out the equations delivers the equality for gradients of bias $\hat{l}$.

$$\begin{aligned}\nabla_{\hat{l}} J_{\text{URAE}}(\pi) &= (y^{(\tau_1)}-x^{(\tau_1)})^T\odot f'(c^{(l(\tau_1))})\\ &= \lambda(\tau_1)\odot f'(c^{(\tau_1)})\\ &= \sum_{\tau\in\mathcal{P}}\lambda(l(\tau))\odot f'(c^{(l(\tau))})\end{aligned} \tag{6.39}$$

For left weights $\hat{L}$ it works the same way.

$$\begin{aligned} D_{\hat{L}} J_{\text{URAE}}(\pi) &= \left(\left(y^{(\tau_1)} - x^{(\tau_1)}\right) \odot f'(c^{(\tau_1)})\right) \otimes y^{(\pi)} \\ &= \left(\lambda(\tau_1) \odot f'(c^{(\tau_1)})\right) \otimes y^{(\pi)} \\ &= \sum_{\tau \in \mathcal{P}(\pi)} \left(\lambda(l(\tau)) \odot f'(c^{(l(\tau))})\right) \otimes y^{(\tau)} \end{aligned} \tag{6.40}$$

It is further assumed that for two arbitrary trees which are rooted by π_1 and π_2, the gradients can be achieved by summing over the weighted terms.

$$\begin{aligned} \nabla_{\hat{L}} J_{\text{URAE}}(\pi_i) &= \sum_{\tau \in \mathcal{P}(\pi_i)} \lambda(l(\tau)) \odot f(c^{l(\tau)}), \text{ for } i = 1, 2 \\ D_{\hat{L}} J_{\text{URAE}}(\pi_i) &= \sum_{\tau \in \mathcal{P}(\pi_i)} \left(\lambda(l(\tau)) \odot f(c^{l(\tau)})\right) \otimes y^{(\tau)}, \text{ for } i = 1, 2 \end{aligned} \tag{6.41}$$

The inductive step is taken by building a new tree with the two previous ones. The new tree has root π and the root's left and right children are $\pi_1 = l(\pi)$ and $\pi_2 = r(\pi)$. The sum of weighted terms can be decomposed into two sums of weights terms belonging to the subtrees of π_1 and π_2 and some term which can be extended by the right node differentiation $D_{\hat{l}} y^{(\pi_2)}$ and $\frac{\partial y_i^{(\pi_2)}}{\partial \hat{L}_{jk}}$ which are in fact simply zero.

$$\begin{aligned} &\sum_{\tau \in \mathcal{P}(\pi)} \lambda(l(\tau)) \odot f'(c^{(l(\tau))}) \\ =& \sum_{\tau \in \mathcal{P}(\pi_1)} \lambda(l(\tau)) \odot f'(c^{(l(\tau))}) + \sum_{\tau \in \mathcal{P}(\pi_2)} \lambda(l(\tau)) \odot f'(c^{(l(\tau))}) + \lambda(l(\pi)) \odot f'(c^{(\pi)}) \\ =& \nabla_{\hat{l}} J(\pi_1) + \nabla_{\hat{l}} J(\pi_2) + \lambda(\pi_1) \odot f'(c^{(\pi_1)}) \\ =& \nabla_{\hat{l}} J(\pi_1) + \nabla_{\hat{l}} J(\pi_2) + \lambda(\pi_1) D_{\hat{l}} y^{(\pi_1)} + \lambda(\pi_2) D_{\hat{l}} y^{(\pi_2)} \\ =& \nabla_{\hat{l}} J(\pi) \end{aligned} \tag{6.42}$$

Carrying out the inductive step for left weights $\hat{L}$ follows the same procedure.

$$\begin{aligned} D_{\hat{L}} J_{\text{URAE}}(\pi) =& D_{\hat{L}} J(\pi_1) + D_{\hat{L}} J(\pi_2) + \left[\sum_i \lambda_i(\pi_1) \frac{\partial y_i^{(\pi_1)}}{\partial \hat{L}_{jk}} + \lambda_i(\pi_2) \frac{\partial y_i^{\pi_2}}{\partial \hat{L}_{jk}}\right]_{jk} \\ =& \sum_{\tau \in \mathcal{P}(\pi_1)} \left(\lambda(l(\tau)) \odot f'(c^{(l(\tau))})\right) \otimes y^{(\tau)} + \sum_{\tau \in \mathcal{P}(\pi_2)} \left(\lambda(l(\tau)) \odot f'(c^{(l(\tau))})\right) \otimes y^{(\tau)} \\ &+ \left(\lambda(\pi_1) \odot f'(c^{(\pi_1)})\right) \otimes y^{(\pi)} \\ =& \sum_{\tau \in \mathcal{P}(\pi)} \left(\lambda(l(\tau)) \odot f'(c^{(l(\tau))})\right) \otimes y^{(\tau)} \end{aligned} \tag{6.43}$$

For gradients of the right parameters, the right branch is taken instead of the left one. Weights $\lambda(\tau)$ are the same as used for left decoding parameters.

$$\begin{aligned} \nabla_{\hat{r}} J_{\text{URAE}}(\pi) &= \sum_{\tau \in \mathcal{P}(\pi)} \lambda(r(\tau)) \odot f'(c^{(r(\tau))}) \\ D_{\hat{R}} J_{\text{URAE}}(\tau) &= \sum_{\tau \in \mathcal{P}(\pi)} \left(\lambda(r(\tau)) \odot f'(c^{(r(\tau))})\right) \otimes y^{(\tau)} \end{aligned} \tag{6.44}$$

Gradients for encoding parameters b, L and R can be formulated simpler in the unfolding scenario than in the standard scenario due to two facts. Full recursion allows to express coefficients in a much easier way. The second reason is that the objective function $J_{\text{URAE}}(\pi)$

for a subtree with root π uses directly just leaf representations from which the actual representations are constant over any parameter and only reconstructed representations show dependency.

$$\begin{aligned}
\frac{\partial J_{\text{URAE}}(\pi)}{\partial b_j} &= \sum_{\tau\in\mathcal{L}(\pi)} \sum_i \left(y_i^{(\tau)} - x_i^{(\tau)}\right) \frac{\partial y_i^{(\tau)}}{\partial b_j} \\
\frac{\partial J_{\text{URAE}}(\pi)}{\partial L_{jk}} &= \sum_{\tau\in\mathcal{L}(\pi)} \sum_i \left(y_i^{(\tau)} - x_i^{(\tau)}\right) \frac{\partial y_i^{(\tau)}}{\partial L_{jk}} \\
\frac{\partial J_{\text{URAE}}(\pi)}{\partial R_{jk}} &= \sum_{\tau\in\mathcal{L}(\pi)} \sum_i \left(y_i^{(\tau)} - x_i^{(\tau)}\right) \frac{\partial y_i^{(\tau)}}{\partial R_{jk}}
\end{aligned} \tag{6.45}$$

In the decoding subtree, no encoding parameters appear directly. It is therefore that no contribution towards any gradient of encoding parameters is performed in the decoding subtree. It, however, has an indirect dependency since the representation $y^{(\pi)}$ of the root node π refers to the encoded representation $x^{(\pi)}$. Hence, only coefficients, aka weights, are derived from the decoding subtree. For simplicity, only bias b is used to derive the coefficients' definition. For encoding weights L and R, the derivation works analogously.

$$\begin{aligned}
\frac{\partial y_i^{(l(\tau))}}{\partial b_j} &= \frac{\partial y_i^{(l(\tau))}}{\partial c_i^{(l(\tau))}} \left(\sum_k \hat{L}_{ik} \frac{\partial y_k^{(\tau)}}{\partial b_j}\right) \\
\frac{\partial y_i^{(r(\tau))}}{\partial b_j} &= \frac{\partial y_i^{(r(\tau))}}{\partial c_i^{(r(\tau))}} \left(\sum_k \hat{R}_{ik} \frac{\partial y_k^{(\tau)}}{\partial b_j}\right)
\end{aligned} \tag{6.46}$$

As seen with the decoding weights, recursion merges two nodes $l(\tau)$ and $r(\tau)$ together into one node τ whilst both bringing along some coefficients $\mu(l(\tau))$ and $\mu(r(\tau))$.

$$\begin{aligned}
&\sum_i \mu_i(l(\tau)) \frac{\partial y_i^{(l(\tau))}}{\partial b_j} + \mu_i(r(\tau)) \frac{\partial y_i^{(r(\tau))}}{\partial b_j} \\
=&\sum_k \sum_i \mu_i(l(\tau)) \frac{\partial y_i^{(l(\tau))}}{\partial c_i^{(l(\tau))}} \hat{L}_{ik} \frac{\partial y_k^{(\tau)}}{\partial b_j} + \mu_i(r(\tau)) \frac{\partial y_i^{(r(\tau))}}{\partial c_i^{(r(\tau))}} \hat{R}_{ik} \frac{\partial y_k^{(\tau)}}{\partial b_j} \\
=&\sum_k \left[\left(\mu(l(\tau)) \odot f'(c^{(l(\tau))})\right) \hat{L} + \left(\mu(r(\tau)) \odot f'(c^{(r(\tau))})\right) \hat{R}\right]_k \frac{\partial y_k^{(\tau)}}{\partial b_j} \\
=&\sum_k \mu_k(\tau) \frac{\partial y_k^{(\tau)}}{\partial b_j}
\end{aligned} \tag{6.47}$$

Coefficients $\mu(\tau)$ follow the same rules and are initialised the same way as coefficients $\lambda(\tau)$ for decoding parameters, that is they are identical. In the follow-up, $\lambda(\tau)$ replaces $\mu(\tau)$ to indicate coefficients of encoding parameters' gradients. The gradient $\nabla_b J_{\text{URAE}}(\pi)$ can be reduced to a single dependency of its root's representation $y^{(\pi)}$.

$$\nabla_b J_{\text{URAE}}(\pi) = \lambda(\pi) D_b y^{(\pi)} = \lambda(\pi) D_b x^{(\pi)} \tag{6.48}$$

The same goes for encoding left and right weights, that is L and R.

$$\begin{aligned}
D_L J_{\text{URAE}}(\pi) &= \left[\sum_i \lambda_i(\pi) \frac{\partial y_i^{(\pi)}}{\partial L_{jk}}\right]_{jk} = \left[\sum_i \lambda_i(\pi) \frac{\partial x_i^{(\pi)}}{\partial L_{jk}}\right]_{jk} \\
D_R J_{\text{URAE}}(\pi) &= \left[\sum_i \lambda_i(\pi) \frac{\partial y_i^{(\pi)}}{\partial R_{jk}}\right]_{jk} = \left[\sum_i \lambda_i(\pi) \frac{\partial x_i^{(\pi)}}{\partial R_{jk}}\right]_{jk}
\end{aligned} \tag{6.49}$$

The gradient $\nabla_b J$ of the overall objective function now can be described with only encoded representations.

$$\nabla_b J_{\text{URAE}} = \sum_{\tau \in \mathcal{P}} \nabla_b J_{\text{URAE}}(\tau) = \sum_{\tau \in \mathcal{P}} \lambda(\tau) D_b x^{(\tau)} \tag{6.50}$$

Similar to expressing gradient contributions in the encoding tree for standard recursive autoencoders, the contributions towards encoding parameters for unfolding recursive autoencoders can be recursively formulated as well. The contribution ρ towards gradient $\nabla_b J_{\text{URAE}}$ from node τ is influenced by parent node $p(\tau)$ with $\gamma D_b x^{(\tau)}$ and by the decoding subtree with $\lambda(\tau) D_b x^{(\tau)}$.

$$\begin{aligned}
\rho =& \gamma D_b x^{(\tau)} + \lambda(\tau) D_b x^{(\tau)} \\
=& (\gamma + \lambda(\tau)) \left[\frac{\partial x_i^{(\tau)}}{\partial a_i^{(\tau)}} \left(\sum_k L_{ik} \frac{\partial x_k^{(\tau)}}{\partial b_j} + R_{ik} \frac{\partial x_k^{(\tau)}}{\partial b_j} + \delta(i,j) \right) \right]_{ij} \\
=& (\gamma + \lambda(\tau)) \odot f'(a^{(\tau)}) + \left[(\gamma + \lambda(\tau)) \odot f'(a^{(\tau)}) \right] \cdot L \cdot D_b x^{(\tau)} \\
&+ \left[(\gamma + \lambda(\tau)) \odot f'(a^{(\tau)}) \right] \cdot R \cdot D_b x^{(\tau)}
\end{aligned} \tag{6.51}$$

Thus, a recursive procedure $\sigma_b(\tau, \gamma)$ can be defined which goes top-down in the encoding tree and increments the gradient $\nabla_b J_{\text{URAE}}$ accordingly.

$$\begin{aligned}
&\sigma_b(\tau, \gamma) = \begin{cases} 0, & \text{if } \tau \text{ is leaf} \\ z_o(\tau,\gamma) + \sigma_b(l(\tau), z_l(\tau,\gamma)) + \sigma_b(r(\tau), z_r(\tau,\gamma)), & \text{else} \end{cases} \\
&\text{with} \\
&\quad z_o(\tau,\gamma) = (\gamma + \lambda(\tau)) \odot f'(a^{(\tau)}) \\
&\quad z_l(\tau,\gamma) = (\gamma + \lambda(\tau)) \odot f'(a^{(\tau)}) \cdot L \\
&\quad z_r(\tau,\gamma) = (\gamma + \lambda(\tau)) \odot f'(a^{(\tau)}) \cdot R
\end{aligned} \tag{6.52}$$

The recursive function $\sigma_b(\pi, \gamma)$ effectively computes the sum of partial gradients of parent nodes in the subtree rooted by π plus the differentiation $D_b x^{(\pi)}$ weighted by γ.

$$\sigma_b(\pi, \gamma) = \gamma \cdot D_b x^{(\pi)} + \sum_{\tau \in \mathcal{P}(\pi)} \nabla_b J_{\text{URAE}}(\tau) \tag{6.53}$$

Starting with a simple tree rooted by π and having two leaf nodes $\tau_1 = l(\pi)$ and $\tau_2 = r(\pi)$, the equation is trivial.

$$\begin{aligned}
\sigma_b(\pi, \gamma) &= (\gamma + \lambda(\pi)) \odot f'(a^{(\pi)}) \\
&= \gamma \odot f'(a^{(\pi)}) + \lambda(\pi) \odot f'(a^{(\pi)}) \\
&= \gamma \cdot D_b x^{(\pi)} + \nabla_b J(\pi) \\
&= \gamma \cdot D_b x^{(\pi)} + \sum_{\tau \in \mathcal{P}(\pi)} \nabla_b J_{\text{URAE}}(\tau)
\end{aligned} \tag{6.54}$$

Furthermore, we assume two trees rooted by π_1 and π_2. The trees can be arbitrary. Starting σ_b at root node π_1 or π_2 is assumed to deliver the sum of parent nodes in the particular subtree plus the differentiation over the root node weighted by γ.

$$\sigma_b(\pi_i, \gamma) = \gamma \cdot D_b x^{(\pi_i)} + \sum_{\tau \in \mathcal{P}(\pi_i)} \nabla_b J_{\text{URAE}}(\tau) \tag{6.55}$$

To carry out the inductive step, another tree which puts together the two previous ones is assumed. It has root node π and the root's child nodes are $\pi_1 = l(\pi)$ and $\pi_2 = r(\pi)$.

Going one level down in the recursion $\sigma_b(\pi, \gamma)$ allows to regroup such that the assumed equation on the root node's level π is proven.

$$\begin{aligned}
\sigma_b(\pi, \gamma) =& z_o(\pi, \gamma) + \sigma_b(pi_1, z_l(\pi, \gamma)) + \sigma_b(\pi_2, z_r(\pi, \gamma)) \\
=& z_o(\pi, \gamma) + z_l(\pi, \gamma) D_b x^{(\pi_1)} + \sum_{\tau \in \mathcal{P}(\pi_1)} \nabla_b J_{\text{URAE}}(\tau) \\
& + z_r(\pi, \gamma) D_b x^{(\pi_2)} + \sum_{\tau \in \mathcal{P}(\pi_2)} \nabla_b J_{\text{URAE}}(\tau) \\
=& (\gamma + \lambda(\pi)) \odot f'(a^{(\pi)}) \left[E + L \cdot D_b x^{(\pi_1)} + R \cdot D_b x^{(\pi_2)} \right] \\
& + \sum_{\tau \in \mathcal{P}(\pi)} \nabla_b J_{\text{URAE}}(\tau) - \nabla_b J_{\text{URAE}}(\pi) \\
=& \gamma \cdot D_b x^{(\pi)} + \lambda(\pi) \cdot D_b x^{(\pi)} + \sum_{\tau \in \mathcal{P}(\pi)} \nabla_b J_{\text{URAE}}(\tau) - \nabla_b J_{\text{URAE}}(\pi) \\
=& \gamma \cdot D_b x^{(\pi)} + \sum_{\tau \in \mathcal{P}(\pi)} \nabla_b J_{\text{URAE}}((\tau)
\end{aligned} \tag{6.56}$$

The recursive functions $\sigma_L(\pi, \gamma)$ and $\sigma_R(\pi, \gamma)$ for left and right encoding weights L and R follow the scheme of the recursive accumulation $\sigma_b(\pi, \gamma)$ but with different coefficients in the non-recursive term.

$$\begin{aligned}
\sigma_L(\tau, \gamma) &= \begin{cases} 0, & \text{if } \tau \text{ is leaf} \\ \hat{z}_o(\tau, \gamma) + \sigma_L(l(\tau), z_l(\tau, \gamma)) + \sigma_L(r(\tau), z_r(\tau, \gamma)), & \text{else} \end{cases} \\
\text{with} & \\
& \hat{z}_o(\tau, \gamma) = (\gamma + \lambda(\tau)) \odot f'(a^{(\tau)}) \otimes x^{(l(\tau))}
\end{aligned} \tag{6.57}$$

$$\begin{aligned}
\sigma_R(\tau, \gamma) &= \begin{cases} 0, & \text{if } \tau \text{ is leaf} \\ \tilde{z}_o(\tau, \gamma) + \sigma_R(l(\tau), z_l(\tau, \gamma)) + \sigma_R(r(\tau), z_r(\tau, \gamma)), & \text{else} \end{cases} \\
\text{with} & \\
& \tilde{z}_o(\tau, \gamma) = (\gamma + \lambda(\tau)) \odot f'(a^{(\tau)}) \otimes x^{(r(\tau))}
\end{aligned} \tag{6.58}$$

6.1.3 Topology

There are two ways to obtain a tree topology as discussed before. Using a syntax parser like the Stanford parser[28] gives a description of the grammatical hierarchy. The trees have no fixed degree, that is parent nodes have arbitrary amount of child nodes. The advantage of getting a syntax tree beforehand is that it removes additional free parameters out of the learning system. Otherwise different topologies have to be tried with an heuristic objection which does not certainly guarantee a correct grammatical description of the sentence. Since the recursive autoencoder takes exactly two nodes to encode a new parent node, the syntax tree has to be transformed into a binary one. It most definitely should exploit the information it has been given from the syntax tree. Nodes which have been lower in the hierarchy than other nodes should stay lower in the binary tree. Nodes which have been higher in the hierarchy than other nodes should stay higher in the binary tree as well. These constraints can be kept by rearranging only sibling nodes when there are not exactly two. In case of only one child node, the parent node is replaced by this child node. If there are more than two child nodes, the binary constraint can be approached on different ways. A heuristic can be applied which - on the curren stand of learning - suggests a solution. In training, the stand of learning can vary and hence the heuristic can vary which can cause counter-productive learning. The other way is to use some transformation which does not change over training because it does not rely on the current stand of learning. A constant transformation should theoretically improve training by shrinking down the space of possibilities and errorneous decisions. A simple transformation is to fold child nodes from the left to the right or vice versa.

The second way of deriving a topology for an input sentence is to create it on the fly. To do so, information in the system help to decide which two nodes are encoded into a new parent node. The heuristic is to achieve a tree topology for which the overall reconstruction distance is minimal[28]. The heuristic assumes that a good system knows grammar that well to assign minimal reconstruction error to the topology which best represents the grammatical hierarchy of the sentence. Socher et al suggest in a different work[29] to use a greedy algorithm to gain such a tree with minimal reconstruction error.
Starting with an ordered list filled with the input nodes, two neighbouring nodes which have minimal reconstruction distance are merged together. The selected nodes are removed from the list and replaced by the newly encoded parent node of these nodes. This procedure iterates until only one node is held in the list. A rough description of this greedy prediction is given in 2. A more detailed view into predicting a topology is given in the next section

Algorithm 2 Greedy Topology Prediction

```
Require: $nInput : Index > 0$
Require: $select : (Index, Index) \to Index$
Require: $score : (Index, Index) \to Index$
  $children : [Index] \leftarrow [0..(nInput - 1)]$
  $merges : [(Index, Index, Index)] \leftarrow []$
  while $children.size > 1$ do
    $\tilde{n} \leftarrow children.size - 1$
    $scores \leftarrow map\ (i \to (i, score(children[i], children[i+1])))\ [0..\tilde{n}]$
    $(i, s) \leftarrow min\ ((i, s) \to s)\ scores$
    $\hat{i} \leftarrow select(children[i], children[i+1])$
    $merges \leftarrow merges + [(children[i], children[i+1], \hat{i})]$
    $children \leftarrow children[0..i-1] + [\hat{i}] + children[i+1..\tilde{n}]$
  end while
  return $merges$
```

about implementation.

6.1.4 Ill-conditioning

Getting a neural network trained to an objective function brings along some hurdles which are mostly due to numerical reason. Given the complex nature of the function which the neural network tries to learn, slow convergence or local minima can lead to non-optimal outcome. It is quite the opposite to memorising the training set rather than generalising from the training set. We can also say that a neural net may tend to generalise too much, that is it stops the learning process too early. Besides some noisy or uncomplete training set, it can be caused by the architecture and initial configuration of the neural network itself, called *ill-conditioing.* Precautions can be met to minimise these issues.

6.1.4.1 Input normalising

Depending on the architecture of the neural network, different input and/or output normalisation are recommended to be applied. In the recursive autoencoder architecture, the objective function is the reconstruction error. Hence, standardising, that is mean of zero and variance of one, will not be applied. To ensure normalised input for the succeding layer, min-max normalisation is applied to transform input into the range $[-1; 1]$. This will speed up the learning time due to numerical reasons explained in the next paragraph about weight initialisation.

6.1.4.2 Weight initialisation

Weights of a neural nets are usually initialised with some uniform distribution over some range $[-r; r]$. The intention is to provide some white noise in the weights which should allow to start at a position which hopefully will lead to a good local , if not global minimum. It is crucial to have proper weights. If weights are too small, activation signals will die propagating through the layers. On the other hand, big weights will saturate activation functions and their derivatives will tend close to zero which will block backpropagating error signals. Latter one is refered to as *paralysis*.

To keep activation and backpropagating error signals from the two extrem situations, normalisation must be applied such that signals stay in range. With aplying the previous paragraph, we assume a normalised input layer. Furthermore, we assume independence between the different input dimensions for simplicity. Our aim is to gain approximate normalisation in the outcome of hidden units. The variance of some hidden unit's weighting net_i can be bounded by the cardinality of node set A_i with an incoming connection to this hidden unit.

$$Var(net_i) \approx \sum_{j \in A_i} Var(w_{ij} y_j) = \sum_{j \in A_i} w_{ij}^2 Var(y_j) \approx \sum_{j \in A_i} w_{ij}^2 \leq |A_i| \, r^2 \tag{6.59}$$

For normalisation with variance 1, the range bound r can be determined accordingly.

$$r_i = \frac{1}{\sqrt{|A_i|}} \tag{6.60}$$

For layers which do not have saturating activation functions, such weight initialisation surely makes sense to prevent any overflow. Although, normalised output has to be guaranteed on different ways like with standard normalisation or softmax.

6.1.4.3 Local learn rate

Like the architecture influences how the activation signals propagate through the net, it also influences the backpropagating error signal. The variance of the error signal δ_j can be bounded by the weighted sum of error signals from the layer P_j behind.

$$Var(\delta_j) \approx \sum_{i \in P_j} Var(\delta_i w_{ij}) = \sum_{i \in P_j} Var(\delta_i) w_{ij}^2 \leq \sum_{i \in P_j} \frac{Var(\delta_i)}{|A_i|} \tag{6.61}$$

The variance of the error signal can be recursively computed.

$$v_j = \begin{cases} \frac{1}{|A_o|} & j \text{ is output} \\ \frac{1}{|A_i|} \sum_{i \in P_j} v_i & \text{else} \end{cases} \tag{6.62}$$

Due to the normalisation of activations, gradient for weights w_{ij} will be expected to scale with the square root of the variance of the respective error signal. Eventually, the characteristic size of the weight, r_i, needs to be considered:

$$\mu_i = \frac{\mu}{|A_i| \sqrt{v_i}} \tag{6.63}$$

6.1.5 Implementation

Because mathematical models and textual descriptions put light on only some issues which come across the implementation, more details will be given here. For some problems, appropriate libraries already exist, like efficient algebraic calculations. Other areas need more work in order to obtain a feasable solution.

6.1.5.1 Algebraic routines

As shown in the section about mathematical models, most computation are of algebraic nature, that is linear operations on matrices and vertices. Various libraries exist which help to make efficient use of different techniques on the CPU. They as well offer simple-to-use interfaces and partly also exploit meta-programming to find better suited rearrangements of slightly more complex terms. For our implementation, we decided to go with blaze[1]. Amongst other things, it gives a portable implementation, it takes use of parallelism with OpenMP and thread library of the C++11 standard and is based on BLAS libraries which stands for Basic Linear Algebra Subprogramms and offers CPU-efficient methods to carry out specific mathematical operations.
Another advantage of using a widely used library is to be ensured that this library is well and intensively tested. Developing such library on its own carries the risk of many small bugs, especially index related ones.

6.1.5.2 Topology

To carry out the right recursions in the neural network, information about the topology need be easily accessible. Several requirements will help to find a suitable approach:

- unfolds and folds arbitrarily within the node
- build routine based on the greedy topology prediction
- independence of actual nodes' representation instances
- simple representations of the topology's state

The independence of the actual nodes' representation instances should help to minimise the coupling to other logical domains within the software. Hence, tests are easier to formulate and to carry out and hopefully bugs are reduced drastically. Template-programming gives a method to achieve some independence from the actual object while still expressing the object itself. We would rather use a complete independent representation and therefore choose to use indices. As tree topologies can be handled on its own without knowing what nodes actually represent, so should the implementation of tree topologies. Any action then is performed with help of these indices which in fact are integers. This way, the topology implementation does not have to hold the actual representation instances and so on, defining a topology and holding the representations are separated as well. To make fulfillment of the remaining requirements easier, the arrangement of the indices need to be carefully designed. Since the indices should allow for access of the nodes' content, indices for the input nodes start by design at zero and go up incrementally, that is the i-th input node will have index $i - 1$ (first input node has index 0). Indices of parent nodes are incremented as well in the order of their merge. The reason for this is discussed later on. The index of the parent node to be merged firstly, is n where n is the amount of input nodes and consequently, the root node will have index $2n - 1$.
The main operations on a tree topology is unfold and fold from arbitrary nodes. To keep the folding order consistent, only nodes which are input nodes or already merged parent

[1]https://code.google.com/p/blaze-lib/

nodes should be used in a merge. For unfolding procedures, the opposite order is required, that is starting from the specified node, only nodes whose parents have been unfolded, can be unfolded. The logical equivalents are recursive functions which ensures that a consistent order is kept. Folding is done in post-order traversal which ensures that the tree or any subtree is traversed bottom-up: Unfolding is done in pre-order traversal to make sure that

Algorithm 3 Post-traversal folding

```
Function fold (n, f):
if $n$ is parent then
    fold(left($n$), $f$)
    fold(right($n$), $f$)
    $f$(left($n$), right($n$), $n$)
end if
```

the tree is traversed top-down: Various folding and unfolding operations are offered to

Algorithm 4 Pre-traversal unfolding

```
Function unfold (n, f):
if $n$ is parent then
    $f$(left($n$), right($n$), $n$)
    fold(left($n$), $f$)
    fold(right($n$), $f$)
end if
```

allow for folding and unfolding over the whole tree or to start an unfolding of some subtree with subsequently iterating over the leaf nodes of the subtree and optionally continuing with folding back to the root node of the subtree. To allow for easy design of combinations of these operations, any information about the current state of unfolding or folding is not part of the state of the topology itself, but encapsulated into methods. To reduce coupling further along, any action in the folding or unfolding process must be generic. Therefore, a function has to be defined as parameter for each folding and unfolding method. On each fold or unfold, this function is called with the respective indices.
Another aim is to keep the state of tree topologies simple in structure and easy in use. Folding and unfolding methods as main operations should be carried out easily and no complex data structure is required since nodes are represented by indices. All in all, two arrays describe a tree topology. The first array stores all the left child nodes, aka the respective indices, and the second array stores all the right child nodes, aka the respective indices. Any two nodes from both arrays at the same position belong together to form a parent node. The order of child nodes goes according to the folding order, that is the first child nodes from both arrays are required to encode the first parent node, the second two child nodes from both arrays are required to encode the second parent node and so on. Folding and unfolding a tree is equivalent to iterate through both arrays at the same time. Any state of folding and unfolding procedure can be described by the position within the array iteration. Folding over the complete tree corresponds to iterate from the first to the last elements, whilst unfolding the complete tree requires to iterate from the last to the first elements. Leafs are characterised by having an index lower than the amount of leafs. Branches have the opposite attribute, that is an index which is at least equal to the amount of leafs. To make folding and unfolding from various positions within the tree possible, an additional constraint has to be set upon the order of the array's elements. Not only should no index $i + n$ until the i-th position in both arrays appear.

$$\forall i = 1 \ldots n - 1 : \forall j < i : l[j] < i + n \land r[j] < i + n \tag{6.64}$$

But subtrees should be also grouped together, that is the previous position should always refer to a child one of the actual node or both child nodes should be leafs.

$$
\begin{aligned}
\forall i = 1 \dots n-1 : &(l[i] < n \wedge r[i] < n) \vee \\
&(l[i] < n \wedge r[i] = i-1+n) \vee \\
&(r[i] < n \wedge l[i] = i-1+n) \vee \\
&(r[i] \geq n \wedge l[i] \geq n \wedge (r[i] = i-1+n \wedge l[i] = i-1+n-j)) \\
&\text{with } j \text{ being the amount of leaf nodes in the right subtree}
\end{aligned}
\tag{6.65}
$$

With these constraints put in place, unfolding from an arbitrary node within the tree works by starting at the respective index position and iterating to the left until no parent node is left which can be checked with simple counting. Folding on only a subtree rooted

Algorithm 5 Subtree unfolding: variable c counts the expected unseen leafs in the subtree; argument i is the index of the subtree root node and f defines a functor which is called with single nodes

```
Function unfoldSubtree ($i$, $f$):
$c = 1$
$n$ = #leafs
while $c > 0$ do
  $l$ = left$[i - n]$
  $l$ = right$[i - n]$
  if $l \leq n$ then
    $c = c + 1$
  end if
  if $r \leq n$ then
    $c = c + 1$
  end if
  $c = c - 1$
  $f(l, r, i)$
  $i = i - 1$
end while
```

by some node with index n has to start from the leftmost leaf node with index $leftLeaf$of this particular subtree. In case of both child nodes being parents, the right subtree is placed next to the root node and further down the left subtree is placed next to the right subtree. This way, folding corresponds to iterating from the position in the left children array where $leftLeaf$ appears until position r. The leftmost leaf node can be either stored

Algorithm 6 Subtree folding: argument l is the index of the left most leaf index in the subtree, r is the index of the root node in the subtree and f defines an action for nodes

```
Function foldSubtree ($l$, $r$, $f$):
$i$ = find $l$ in leftChildren$[0 \dots n]$
while $i \leq r$ do
  $f$(leftChildren$[i]$, rightChildren$[i]$, $i + n$)
  $i = i + 1$
end while
```

or deduced in combination with unfold.
Building a topology based on the previously discussed heuristics will be illustrated here more in depth. Parameters for predicting the topology are the amount of leaf nodes $nLeafs$ and some procedures *select* which marks the next selected merge, *score* which calculates

the score of merging two neighbouring nodes and *sort* which is called at the end of the building procedure to allow for resorting of parent nodes' representations. The building process consists of two subprocesses. The first subprocess aims towards constructing a topology which is conform with the first constraint 6.64. The greedy prediction algorithm 2 is improved to minimise calls of *score* and to reduce memory space. Arrays for left and right child nodes and scores are held whose length is constant over the building process. Initially, left and right child nodes' arrays hold all potential parent combinations and the scores array stores all the respective reconstruction errors. Starting with a range spanning over all elements, on each iteration the left and right nodes with minimal score are moved to the leftmost position in the range while all elements left of this particular position within the range are moved one to the right. The range is moved to the right and scores and child nodes are updated where necessary. The iteration ends if only one candidate is left. Hence only $nLeafs - 1$ iterations are necessary for this iteration. The second subprocess rearranges merges such that subtrees of child nodes are placed directly left of the parent node position. It corresponds to reinforcing the second constraint 6.65. Parent nodes are enumerated incrementally via post-order traversal. This enumeration marks the wanted order of parent merges. Next, an array is instantiated with corresponding indices of left and right child nodes and the position where they should be placed. The array is subsequently sorted by the targeted position. Eventually, indices of the left and right child nodes need to be remapped if they are parent nodes. Lastly, *sort* is called with the new arrangement to allow for repositioning of associated data.

6.1.5.3 RAE

The recursive autoencoder holds matrices and vectors for encoding weights L and R, encoding bias b as well for respective instances for untied architectures. Decoding weights $\hat{L}$ and $\hat{R}$ and biases $\hat{l}$ and $\hat{r}$ for tied and untied architectures are attributes of the recursive autoencoder too.
The main methods are training, testing and encoding. Training can be carried out in batching mode with optional stochastic sampling. For a training set which is small enough, all samples are considered to apply correction on the network parameters. Large training data sets cause long computation times and may potentially lead to numeric overflows. Taking only small batches of samples represent the whole training set with some statistic error. Taking these statistic errors into account, training cycles are computed much faster and numerical problems usually do not arise. To make the system more robust, artificial noise is added by randomly leaving out samples when grouping samples to batches. This way, batches should be different for each training cycle. In our implementation, stochastic sampling can be activated with a parameter for a geometric distribution from which the amount of samples to leave out is drawn. Despite using batches in a recursive autoencoder, numerical overflows still appear. Variable length of the input sentence makes it difficult to find a feasable size for batches. Two methods are implemented to overcome this struggle. The first method is length normalisation. If length normalisation is activated, gradients of one sample will be normalised to a specific length (concatenating the gradients to a vector) in each training batch. All normalised gradients which are free of overflows are summed up and onc again length normalisation is applied, before correcting any parameters. The second method is clipping and works similar to length normalisation. Another name for clipping is max normalisation which refers to the mechanism that normalisation is only applied if the respective length exceeds some threshold. It will be then clipped or scaled down to the specific length. Differences of both methods are that length normalisation is always applied and not only handles values which are too big but also too small. This methods can be used as well for preempting weight explosion and implosion, that is oszillating and vanishing weights. Another method for preventing weight explosion is weight decay, also called l2-Penalty. The idea is to punish parameters which become to large. It

Algorithm 7 Heuristic construction of topology: n is the size of leafs, functor *select* indicates which nodes are merged next, *score* evaluates a potential merge and *sort* can be used to sort associated data to have correctly indices

```
Function build ($n$, $select$, $score$, $sort$):
lefts$[0,\dots,n-1] = [0,\dots,n-1]$
rights$[0,\dots,n-1] = [0,\dots,n-1]$
scores$[0,\dots,n-1] = [score(0,1),\dots,score(n-1,n)]$
for $i = 0$ to $n-1$ do
    $j$ = argmin scores$[k]$ over $k = i,\dots,n-1$
    $l$ = lefts$[j]$
    $r$ = rights$[j]$
    $select(l,r)$
    move lefts$[i,\dots,j-1]$ to lefts$[i+1,\dots,j]$
    move rights$[i,\dots,j-1]$ to rights$[i+1,\dots,j]$
    move scores$[i,\dots,j-1]$ to scores$[i+1,\dots,j]$
    lefts$[i] = l$
    rights$[i] = r$
    if $j+2 < n$ then
        lefts$[j+1] = i+n$
        scores$[j+1] = score(i+n,$rights$[j+1])$
    end if
    if $j > i$ then
        rights$[j] = i+n$
        scores$[j] = score($lefts$[j]$, $n+i)$
    end if
end for
newOrder = []
$g = [c=n](p)\{$newOrder$[p-n] = c++\}$
traverse post-order lefts and rights with $g$
for $i = 0$ to $n-1$ do
    if lefts$[i] < n$ then
        $l$ = lefts$[i]$
    else
        $l$ = newOrder[lefts$[i]$ $-n$]
    end if
    if rights$[i] < n$ then
        $r$ = rights$[i]$
    else
        $l$ = newOrder[rights$[i]$ $-n$]
    end if
    indexedChildren$[i] = (l,r,i)$
end for
sort indexedChildren by $(a,b)\{$newOrder$[a.i]<$newOrder$[b.i]\}$
$sort$(newOrder)
for $i = 0$ to $n-1$ do
    lefts$[i]$ = indexedChildren$[i].l$
    rights$[i]$ = indexedChildren$[i].r$
end for
return (lefts, rights)
```

is based on the heuristic that information and importance should be distributed roughly equally on the system. Therefore, one parameter should not carry that much impact to make a decision on its own. Numerically speaking, norm of parameter values should be as low as possible. This behaviour can be achieved by extending the objective function with a sum over all weight components' quadrats.

$$J_{\mathrm{l2}} = J + \sum_i \sum_j L_{ij}^2 + R_{ij}^2 + \hat{L}_{ij}^2 + \hat{R}_{ij}^2 \tag{6.66}$$

Only parameters of matrix weights are included whilst biases are omitted. The derivation for a single weight L_{ij} is extended by its own value. If it has a big value, it will be pushed far into the different direction. Small values are not much corrected by l2-Penalty.

$$\frac{\partial J_{l2}}{\partial L_{ij}} = \frac{\partial J}{\partial L_{ij}} + L_{ij} \tag{6.67}$$

Introducing additional noise into the system leads to more robust solutions since the system has learned to expect small variations. Two related techniques are implemented which can be used at the same time. The first technique is input dropping. Each training batch will use a mask to set input components to zero. Which components are set to zero is decided by a Bernoulli distribution. The second technique is called DropOut and applies a vector mask on each activation vector. It is suggested by Srivastava et al [30] and has been shown to enforce good conversion and allowing for more parameters while restraining overfitting. Conversion is achieved by gradient descend. The descend is accelerated or slowed down by the learn rate. It has to be initialised by parameters and without any further options, it will be constant during training. For fast conversion, however, the learn rate should change to the local situation. If the direction of the gradient descend does not change much for some train cycles, fasten it up will help to avoid unnecessarily small steps. If the direction of the gradient descend changes frequently, learn rate should go down to make the search for a minimum more sensitive. A simple approach is to employ a momentum of the previously used descend $d(t)$ for the coming descend $d(t+1)$ with some coefficient α_{mom} and the currently gradient $\Delta w(t+1)$.

$$d(t+1) = \Delta w(t+1) + \alpha_{\mathrm{mom}} \cdot d(t) \tag{6.68}$$

The momentum will cause the the current descend to hold direction to some extend from the previous direction. If the descending direction is about the same for several steps, the steps will get longer. If the direction of gradient changes, the impact of the previous direction will reduce the length of the step. An additional approach which can be used in combination with learn momentum, is learn adaption. The idea is to shrink or extend the learn rate based on how much the descend will change. One way to measure this change is cosine similarity in which the cosine angle θ of two vectors a and b is determined by scalar product and normalisation.

$$cos(\theta) = \frac{a \cdot b}{||a|| \cdot ||b||} \tag{6.69}$$

Cosine similarity gives a value in range from -1 to 1. If the direction is about the same, cosine similarity will be approximately 1. If the direction is about the opposite to the previous one, cosine similarity will be approximately -1. If the direction is about orthogonal to the previous one, cosine similarity will bee about 0. Exploiting this fact, the learning rate $r(t)$ can be adapted by adding an adaption constant κ scaled by the cosine similarity $cos(\theta)$.

$$r(t+1) = r(t) + \kappa \cdot \cos(\theta) \tag{6.70}$$

The effect is that for reverse changes in direction, the learning rate will be reduced, that is descend will slow down. For steering the same direction, the learning rate will be increased,

that is descend will speed up. Orthogonal direction changes will not influence the learning rate. In combination with learn momentum, useful techniques can be put in place to overcome small non-convex regions.
Since neural networks are usually too complex to find a global minimum, gradient descend will search for a local minimum. To obtain a measurement which is independent of the training data, a held-out test set is applied whose average reconstruction error is taken as metric. Whether the system has found a local minimum, can be decided only in hindsight. The regular behaviour of test reconstruction is a slow descend at the beginning which then goes faster down once the system has found a basic configuration. At some point, the system will start to memorise the data rather than learning abstract features. It is marked by an increase in the test reconstruction error. It is this point where the system has the most knowledge with the best generalisation and it should be the final configuration. This approach is called early stopping. To find this configuration, the configuration with minimal test reconstruction error is stored and updated if a configuration with lower test reconstruction error has been found. The training stops after a specified amount of training cycles, when no configuration with lower test reconstruction error has been found. Sometimes the random initialisation of parameters turn out to be not applicable to the objective function which will cause an increase in test error at the beginning until a suitable configuration is found. To prevent early-stopping from terminating the training procedure, a specified amount of cycles are trained without considering termination. Both parameters, the window size for finding a minimal test error and the minimal amount of training cycles, have to be set manually.

6.2 Perplexity

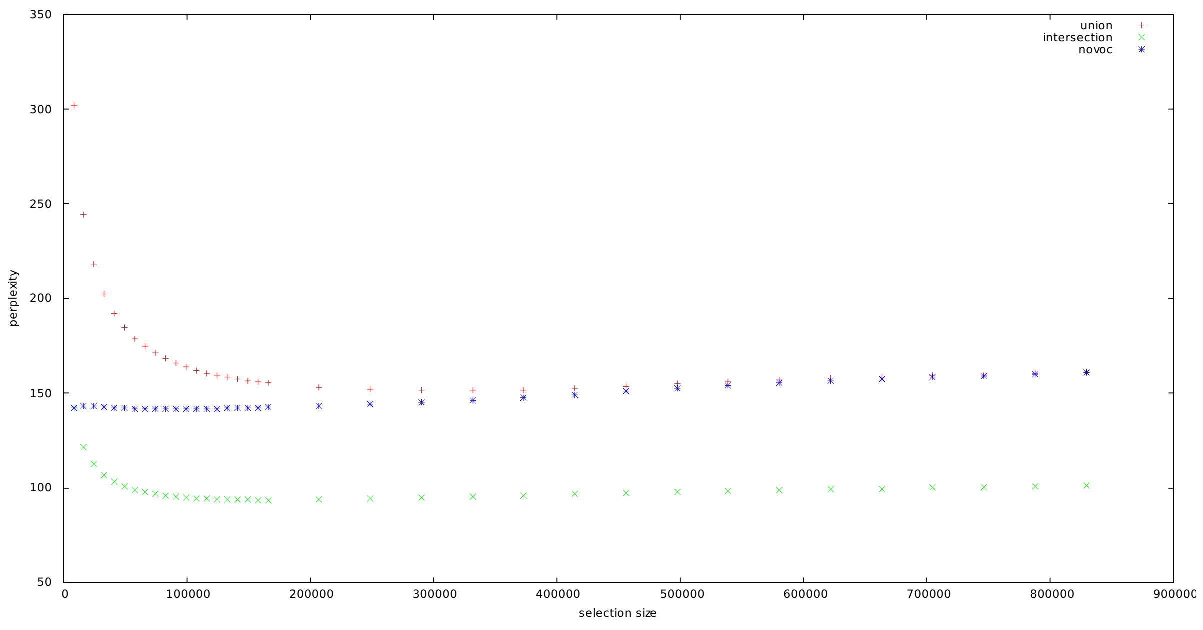

Figure 6.4: Perplexity of selections from Papers for TED in-domain without enrichment; the target of the perplexity is TED close-in-domain; minimal perplexity is reached for intersection at 165 791 (2%) with 93.5783 and for union at 331 582 (3%) with 151.467

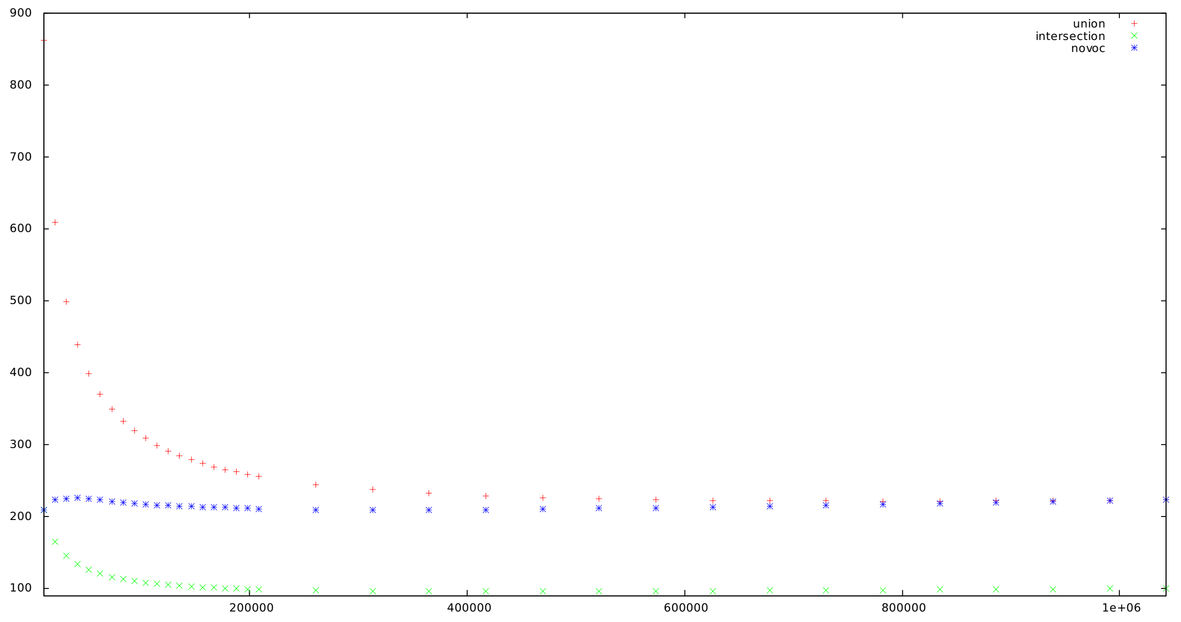

Figure 6.5: Perplexity of selections from Collcetions for TED in-domain without enrichment; the target of the perplexity is TED close-in-domain; minimal perplexity is reached for intersection at 41 714 (4%) with 95.4604 and for union at 782 141 (7.5%) with 221.132

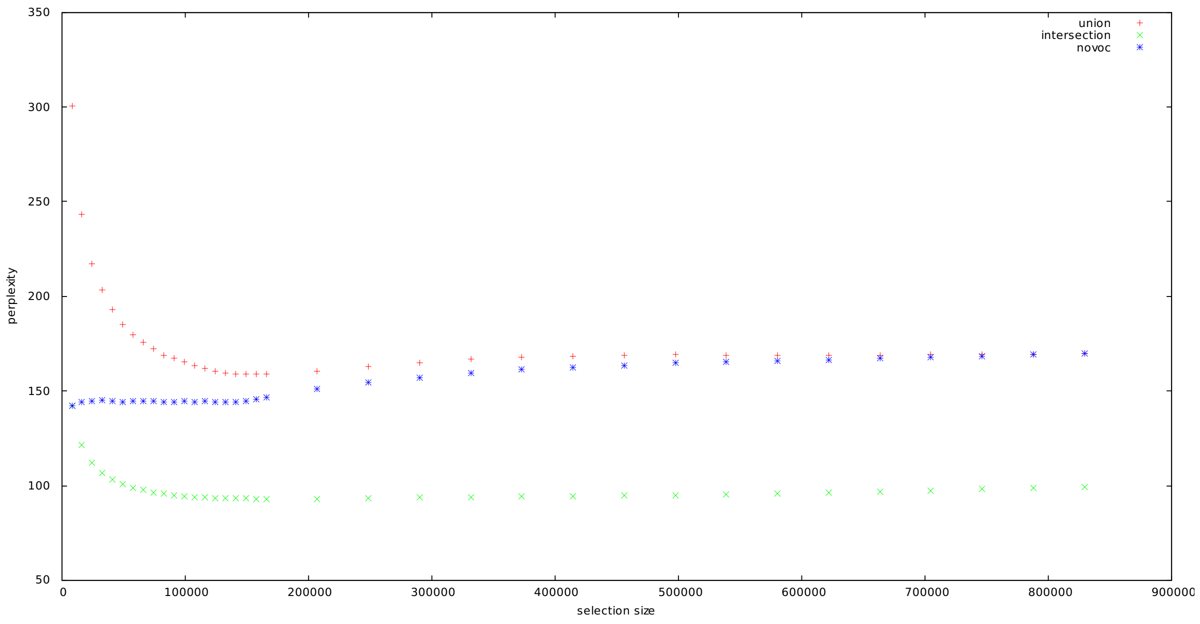

Figure 6.6: Perplexity of selections from Papers for TED in-domain with random replacement (0.2) and n-gram pivot alignment; the target of the perplexity is TED close-in-domain; minimal perplexity is reached for intersection at 165 791 (2%) with 93.5783 and for union at 331 582 (3%) with 151.467

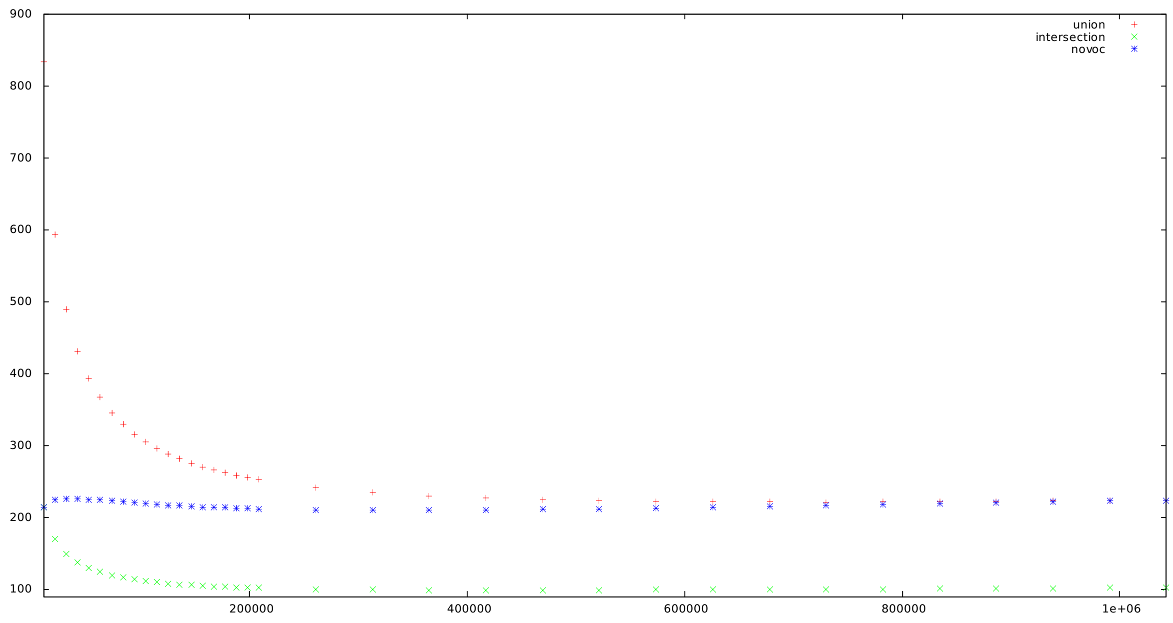

Figure 6.7: Perplexity of selections from Collections for TED in-domain with random replacement (0.2) and n-gram pivot alignment; the target of the perplexity is TED close-in-domain; minimal perplexity is reached for intersection at 165 791 (2%) with 93.5783 and for union at 331 582 (3%) with 151.467

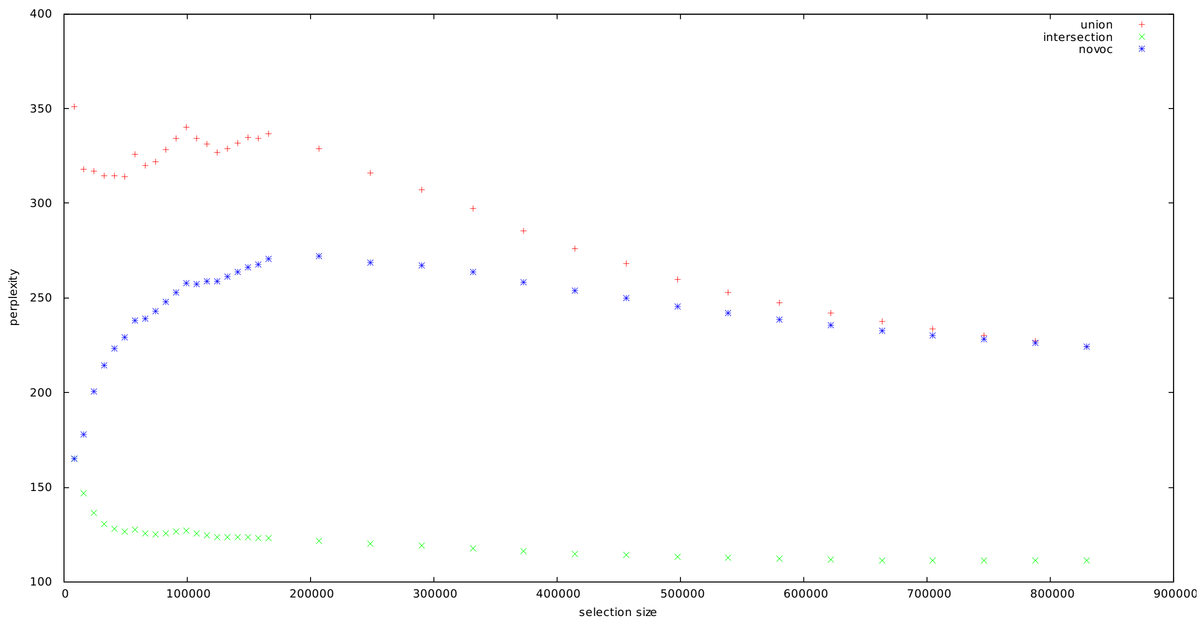

Figure 6.8: Perplexity of selections from Papers for TED in-domain with SMT decoding and n-gram pivot alignment; the target of the perplexity is TED close-in-domain

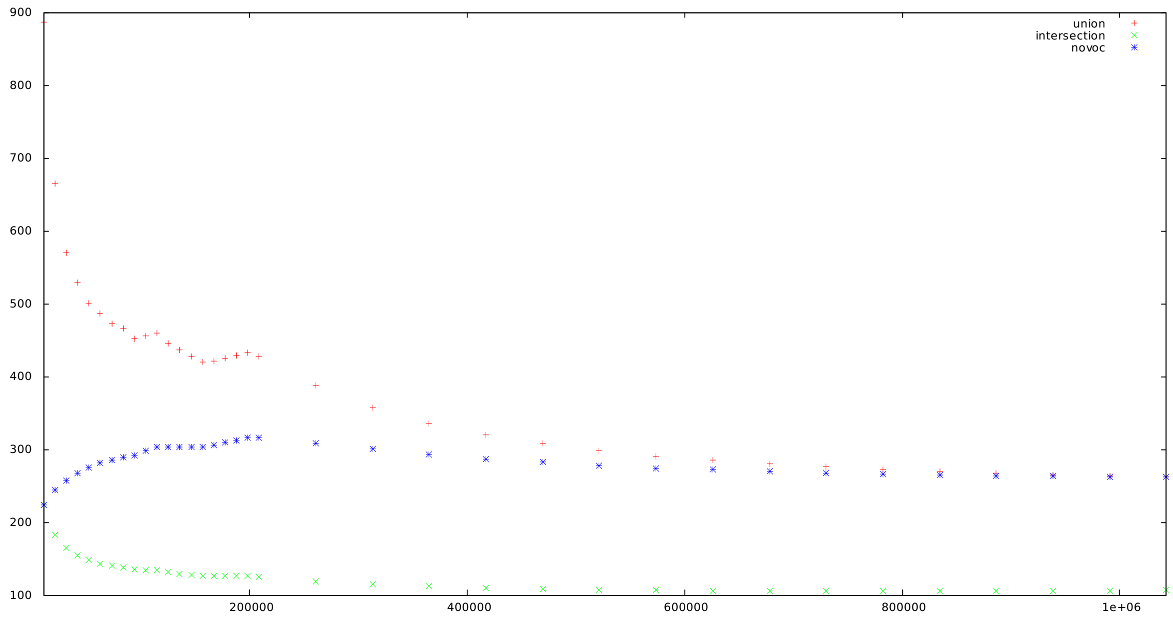

Figure 6.9: Perplexity of selections from Collections for TED in-domain with SMT decoding and n-gram pivot alignment; the target of the perplexity is TED close-in-domain

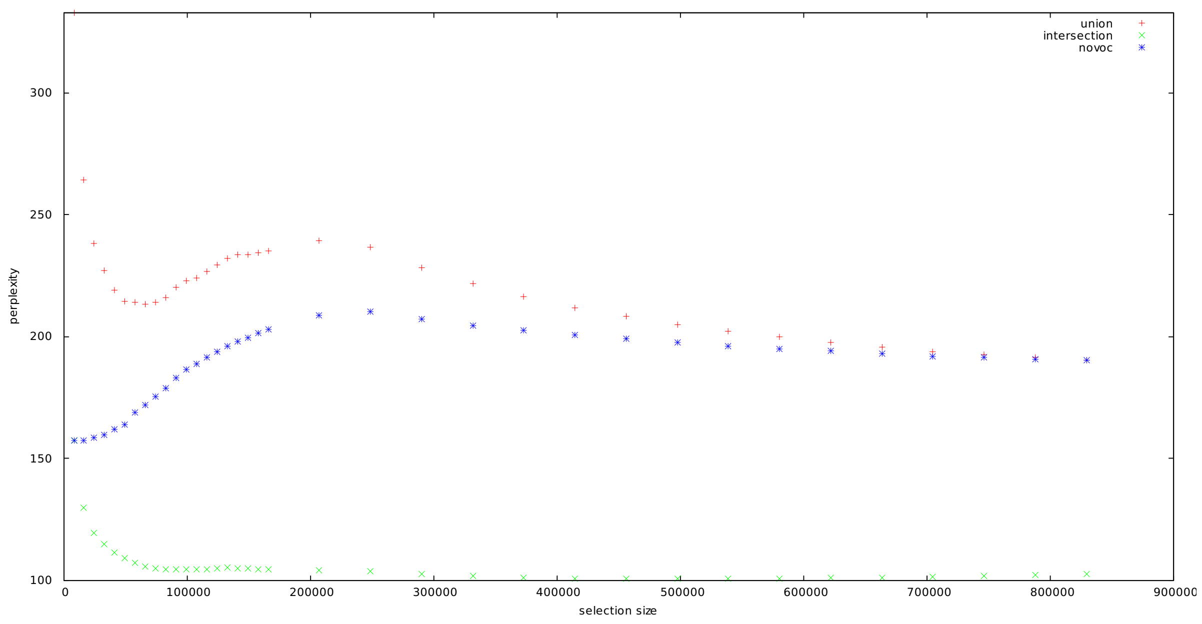

Figure 6.10: Perplexity of selections from Papers for TED in-domain with random replacement (0.2) and RAE; the target of the perplexity is TED close-in-domain

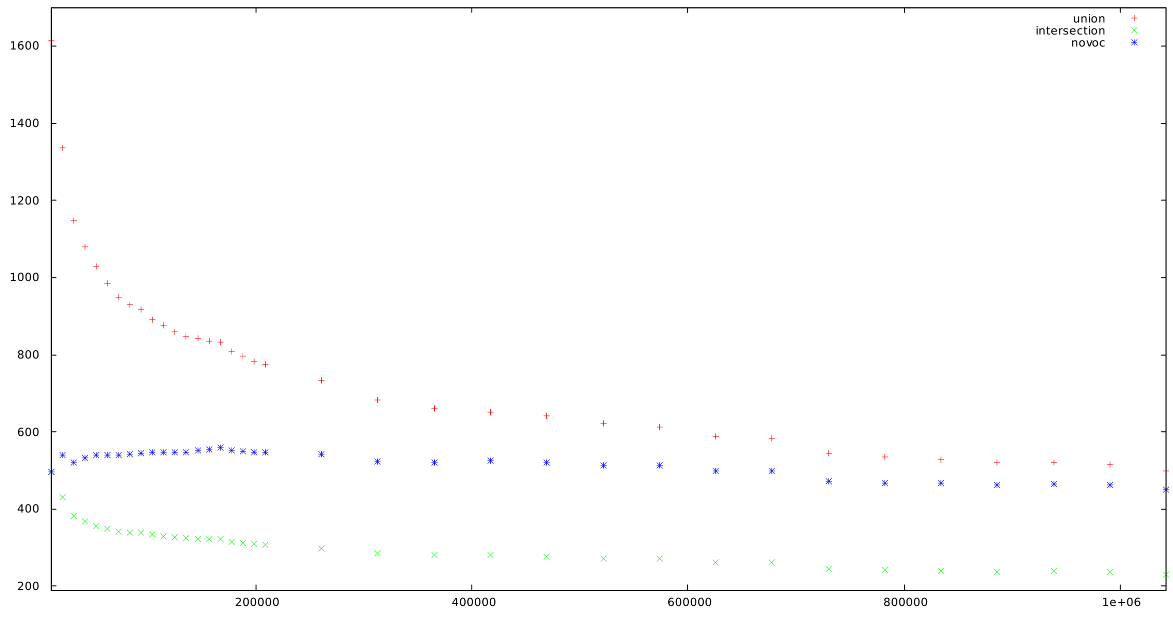

Figure 6.11: Perplexity of selections from Collections for TED in-domain with random replacement (0.2) and RAE; the target of the perplexity is TED close-in-domain

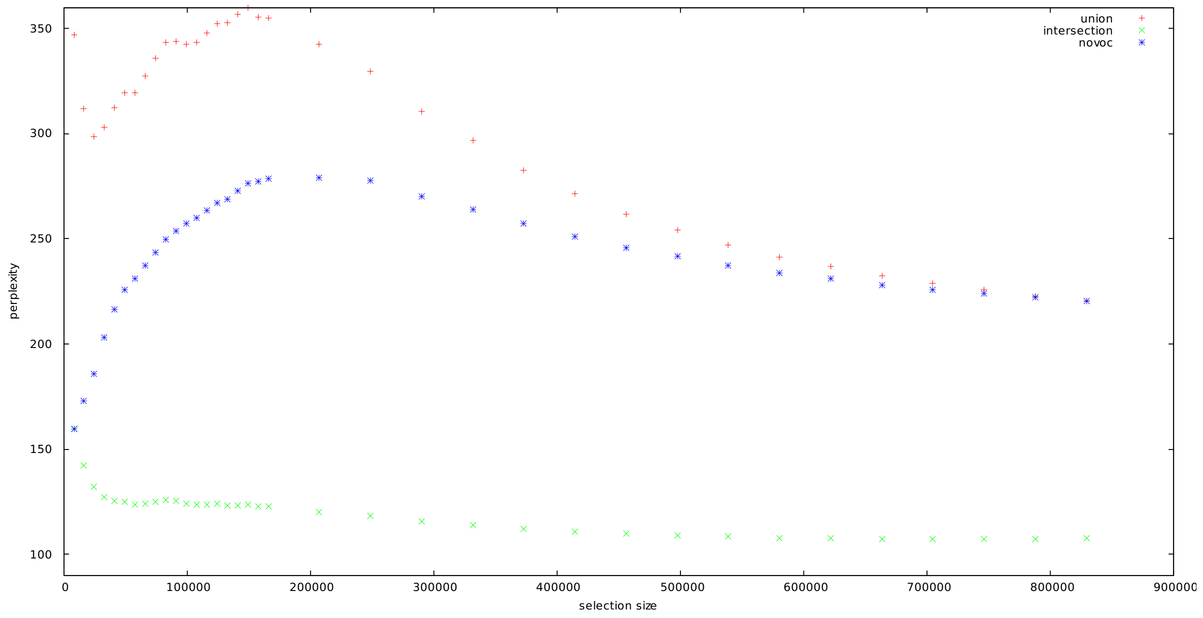

Figure 6.12: Perplexity of selections from Papers for TED in-domain with SMT decoding and RAE; the target of the perplexity is TED close-in-domain

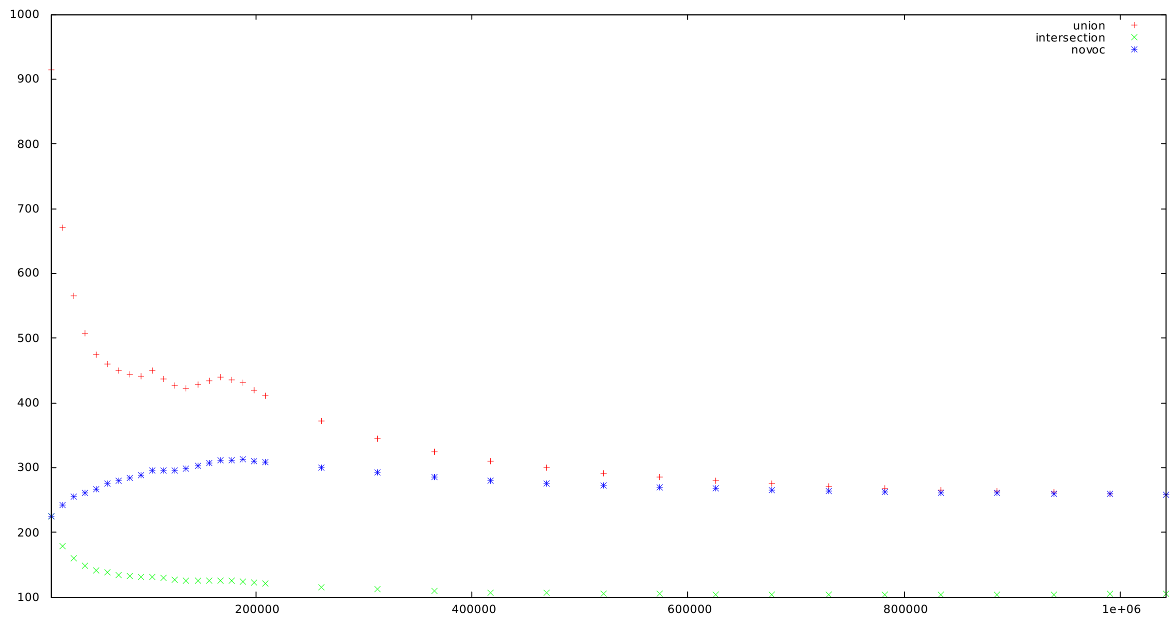

Figure 6.13: Perplexity of selections from Collections for TED in-domain with SMT decoding and RAE; the target of the perplexity is TED close-in-domain

7. Postscriptum

Additionally to the experiments with TED in-domain, a corpus composed of two transcriptions of lectures is used as selection corpus. It is refered to as *Lectures*. According to Mediani et al[20], decreasing the size of the selection corpus can help to achieve not only to get a better BLEU score in general but also compared to the baseline of Moore and Lewis.

Corpus	sentences
Lectures	3417
Lectures tuning	1000
Lectures testing	1500
Lectures selection	700
Lectures selection target	717

Table 7.1: Lectures corpora and their amount of sentences

Equivalent to the experimental procedure with TED in-domain as selection corpus, selection based on perplexity is performed on Papers and Collections according to Moore and Lewis. Only perplexities on intersected vocabulary are considered giving their hardest restriction on the vocabulary. Despite the observation of Mediani et al that reducing the

Configurations	Collections	Papers
Baseline	96.3	71.6332
SMT Alignment	119.25	97.8943
SMT RAE	119.905	101.188
RR 0.2 Alignment	115.449	93.1491
RR 0.5 Alignment	112.426	85.0884
RR 0.9 Alignment	104.878	81.2253
RR 0.2 RAE	114.622	88.654
RR 0.5 RAE	103.157	80.1691
RR 0.9 RAE	106.039	82.1438

Table 7.2: Perplexities of selections for Lectures with intersected vocabulary

size of selection corpus might help to outperform the perplexity achieved by Moore and Lewis, none of it can be observed in this scenario. Corpora produced by SMT methods

Configuration	Collections	Papers
Baseline without selection	16.68	16.68
Baseline with selection	17.26	17.56
Baseline wit entire corpus	17.51	16.82
SMT Alignment	17.10	17.20
SMT RAE	17.20	17.42
RR 0.2 Alignment	17.36	17.41
RR 0.5 Alignment	17.29	17.36
RR 0.9 Alignment	17.29	17.62
RR 0.2 RAE	17.59	17.56
RR 0.5 RAE	17.52	17.47
RR 0.9 RAE	17.55	17.60

Table 7.3: BLEU score on selections for Lectures with minimal perplexity on intersected vocabulary

show higher perplexities compared to corpora produced by Random Replacement. The parameter of distribution appears to have no correlation to the resulting perplexity in Random Replacement.

For extrinsic evaluation, selections with minimal perplexity on intersected vocabulary are taken. The extrinsic evaluation is a BLEU scoring in an SMT system equivalent to the previously used SMT system. Two language models are used for each configuration. The first model is the same used in the other experiments, that is a n-gram model of order four with Knesser-Ney smoothing built from EPPS, NC and TED rest. The second language model is built from the respective selection of the configuration. It is as well a n-gram model of order four with Knesser-Ney smoothing. The translation model is taken as well from the previous experiments. Different to the system of the previous experiments are the corpora for testing and tuning. Due to a different selection corpus, these corpora have to be replaced accordingly. Reducing the size of the selection corpus yields a performance boost. Selecting according to Moore and Lewis on Collections does not bring an advantage compared to using the entire corpus instead. For Papers, carrying out Moore and Lewis gives an enhancement compared to using the entire corpus. Paraphrasing via SMT does not deliver better results than using Moore and Lewis' selection. However, applying random replacement brings some improvement over the baselines for both corpora. For Collections, random replacement works better with the lexicon based on the recursive autoencoder than indirect alignment. For Papers, this difference cannot be seen between using RAE based lexicon and using indirect Alignment based lexicon. It is nevertheless difficult to compare these particular results due to differences in the lexica. On Collections, the improvement over the baseline yields 0.08% and on Papers, the improvement over the baseline yields 0.08% as well.

As seen in previous experiments, the relationship between perplexity and BLEU scoring is not as clear as on might be assuming. For a small selection corpus, it appears that selection by Moore and Lewis does not bring improvement on low noisy corpora, contrary to noisy corpora. Also, perplexities on selections with enriched selection corpora do not correlate with the respective BLEU scores. What can be seen, is the performance increase over the baseline which Mediani et al report. Comparing the paraphrasing methods and the lexica separately, Random replacement seems to perform better than SMT. For the lexica, the data suggests that recursive autoencoders deliver better synonym information than indirect alignment according to Bannard and Callison-Burch. However, to compare the lexica rightfully is not straight forward since thresholds in the postprocessing step for building lexica are found manually.

Bibliography

[1] Amittai Axelrod, Xiaodong He, and Jianfeng Gao. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 355–362, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[2] Colin Bannard and Chris Callison-Burch. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 597–604, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[3] Petra Barancíková and Aleš Tamchyna. Machine translation within one language as a paraphrasing technique. *Vera Kurková, Lukáš Bajer (Eds.)*, page 1.

[4] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003.

[5] Oyvind Raddum Berg, Stephan Oepen, and Jonathon Read. Towards high-quality text stream extraction from pdf: Technical background to the acl 2012 contributed task. In *Proceedings of the ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries*, ACL '12, pages 98–103, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[6] Ondrej Bojar and Zdenek Zabokrtský. Czeng: Czech-english parallel corpus release version 0.5. *Prague Bull. Math. Linguistics*, 86:59–62, 2006.

[7] Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.*, 19(2):263–311, June 1993.

[8] Jonathan Chevelu, Ghislain Putois, and Yves Lepage. The true score of statistical paraphrase generation. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 144–152, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[9] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA, 2008. ACM.

[10] William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*. Asia Federation of Natural Language Processing, 2005.

[11] Tianchuan Du and Vijay K Shanker. Deep learning for natural language processing.

[12] George Foster and Roland Kuhn. Mixture-model adaptation for smt. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 128–135, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[13] Jianfeng Gao, Joshua Goodman, Mingjing Li, and Kai-Fu Lee. Toward a unified approach to statistical language modeling for chinese. 1(1):3–33, March 2002.

[14] Thanh-Le Ha, Teresa Herrmann, Jan Niehues, Mohammed Mediani, Eunah Cho, Yuqi Zhang, Isabel Slawik, and Alex Waibel. The kit translation systems for iwslt 2013. 2013.

[15] Ozan İrsoy and Claire Cardie. Opinion mining with deep recurrent neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 720–728, 2014.

[16] D. Klakow. Selecting articles from the language model training corpus. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on*, volume 3, pages 1695–1698 vol.3, 2000.

[17] Philipp Koehn and Josh Schroeder. Experiments in domain adaptation for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 224–227, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[18] Cornelius Lanczos. An iterative method for the solution of the eigenvalue problem of linear differential and integral, 1950.

[19] Sung-Chien Lin, Chi-Lung Tsai, Lee-Feng Chien, Keh-Jiann Chen, and Lin-Shan Lee. Chinese language model adaptation based on document classification and multiple domain-specific language models. In George Kokkinakis, Nikos Fakotakis, and Evangelos Dermatas, editors, *EUROSPEECH*. ISCA, 1997.

[20] Mohammed Mediani, Joshua Winebarger, and Alexander Waibel. Improving in-domain data selection for small in-domain sets. 2014.

[21] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119, 2013.

[22] Marvin Minsky and Seymour Papert. *Perceptrons - an introduction to computational geometry.* MIT Press, 1987.

[23] Robert C. Moore and William Lewis. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 220–224, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[24] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *AISTATS'05*, pages 246–252, 2005.

[25] Franz Josef Och and Hermann Ney. Improved statistical alignment models. In *In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 440–447, 2000.

[26] Romain Paulus, Richard Socher, and Christopher D Manning. Global belief recursive neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2888–2896. Curran Associates, Inc., 2014.

[27] Chris Quirk, Chris Brockett, and William Dolan. Monolingual machine translation for paraphrase generation. In *In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 142–149, 2004.

[28] Richard Socher and Eric H. Huang and Jeffrey Pennington and Andrew Y. Ng and Christopher D. Manning. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *Advances in Neural Information Processing Systems 24*. 2011.

[29] Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2011.

[30] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[31] Joseph Turian, Département D'informatique Et, Recherche Opérationnelle (diro, Université De Montréal, Lev Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semisupervised learning. In *In ACL*, pages 384–394, 2010.

[32] Alexander Waibel, Toshiyuki Hanazawa, Geofrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. Readings in speech recognition. chapter Phoneme Recognition Using Time-delay Neural Networks, pages 393–404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[33] Haijun Zhang, John K. L. Ho, Q. M. Jonathan Wu, and Yunming Ye. Multidimensional latent semantic analysis using term spatial information. *IEEE T. Cybernetics*, 43(6):1625–1640, 2013.