PRESEMT Disambiguation Report

George Tsatsanifos

February 15, 2013

Contents

1 Introduction **5**

2 SOM-based Disambiguation **5**

2.1 Implementation Details 8

2.2 ABC Analysis 9

2.3 English Monolingual Language Models 10

2.3.1 Standard Model 11

2.3.2 Increasing the Number of Training Iterations 13

2.3.3 Appropriate Configuration of B Category Bounds 20

2.3.4 The Effect of Map Dimensionality 21

2.3.5 Filtering by Tag 25

2.3.6 Reduced Feature Vector 27

2.4 German Monolingual Language Models 29

3 N-gram based Disambiguation **30**

4 Phrase Heads and Functional Heads Processing **34**

5 Conclusions **43**

List of Figures

1 BMUs of a hexagonal SOM topology for two-dimensional datasets. 7
2 Trellis diagram example. 8
3 set C_a using B category bounds 70%-85% for 100 iterations. 12
4 set C_a using B category bounds 70%-85% for 500 iterations. 12
5 Representation of the training process for a set C_a map with B category bounds set to 70%-85% for 0 iterations (initialization). 14
6 Representation of the training process for a set C_a map with B category bounds set to 70%-85% for 100 iterations. 14
7 Representation of the training process for a set C_a map with B category bounds set to 70%-85% for 400 iterations. 15
8 Representation of the training process for a set C_a map with B category bounds set to 70%-85% for 800 iterations. 15
9 Representation of the training process for a set C_a map with B category bounds set to 70%-85% for 1000 iterations. 16
10 Representation of the training process for a set C_a map with B category bounds set to 70%-85% for 1200 iterations. 16
11 set C_c SOM using B category bounds 70%-85% for 480 iterations 17
12 A representation of the disjunctive words over SOM along with the selected route that includes the selected words. 17
13 set C_a SOM using B category bounds 70%-85% for 5000 iterations. 18
14 set C_b SOM using B category bounds 70%-85% for 1000 iterations. 18
15 set C_b SOM using B category bounds 70%-85% for 5000 iterations. 19
16 The Pareto cumulative distribution that the ABC analysis is based on. . . . 19
17 set C_b SOM using B category bounds 80%-85% and 5000 iterations. 21
18 set C_b SOM using B category bounds 70%-75% and 2500 iterations. 22
19 set C_a SOM using B category bounds 70%-75% and 3000 iterations. 22
20 set C_d SOM using B category bounds 70%-75% and 1200 iterations. 23
21 set C_d SOM using B category bounds 70%-75% and 2500 iterations. 23
22 set C_a SOM using B category bounds 70%-75% and 3000 iterations with threshold set to 6. 24
23 set C_a SOM using B category bounds 70%-75% and 3000 iterations with threshold set to 8. 25

24 set C_a SOM using B category bounds 70%-75% and 3000 iterations with threshold set to 10. 26
25 Comparison of different disambiguation methods. 26
26 set C_b using B category bounds 70%-75% and 3000 iterations with dimensionality 15 x 20 28
27 set C_b using B category bounds 70%-75% and 3000 iterations with dimensionality 17 x 38 29
28 set C_b using B category bounds 70%-75% and 3000 iterations with dimensionality 30 x 50 30
29 set C_b SOM using B category bounds 70%-75% and 3000 iterations with filtered 20% of the rarest words 31
30 set C_b SOM using B category bounds 70%-75% and 3000 iterations with filtered 30% of the rarest words 32
31 set C_b SOM using B category bounds 70%-75% and 3000 iterations with filtered 40% of the rarest words 33
32 set C_b SOM using B category bounds 70%-75% and 3000 iterations with filtered 50% of the rarest words 34
33 set C_b SOM using B category bounds 70%-75% and 3000 iterations with filtered 60% of the rarest words 35
34 set C_b SOM using B category bounds 70%-75% and 3000 iterations with filtered 70% of the rarest words 36
35 set C_c German SOM using B category bounds 80%-82% and 600 iterations. 36
36 set C_c German SOM using B category bounds 70%-75% and 480 iterations. 37
37 Training headSOM using English corpus (3GBs) with B category bounds 70%-85% and 1200 iterations. 37
38 Training headSOM using English corpus (3GBs) with B category bounds 70%-75% and 1200 iterations. 38
39 Training headSOM using German corpus with B category bounds 70%-75% and 500 iterations. 39
40 A representation of the disjunctive words over the Kohonen map along with the selected route that includes the selected head-words. 39
41 A representation of the disjunctive words over the Kohonen map along with the selected route. 40

List of Tables

1 SOM evolution scores. 11
2 SOM evolution scores. 13
3 Feature vector cardinality for different configuations. 21
4 Disambiguation results for varying cut-off frequency thresholds for the rarest words of the corpus. 27
5 Results for the 25MB English corpus. 28
6 Disambiguation results for various head-based models. 40
7 Disambiguation results for the SEMEVAL benchmark. 41

1 Introduction

In this work, we study the problem of Word Translation Disambiguation (WTD) which essentially deals with selecting the best possible translations of words, given a sentence in the source language, according to a bilingual dictionary or some other translation model, as a source language word can often have several translations in the target language. We address the problem through two distinct approaches to model the TL language. (i) First, using a novel method which relies on Kohonen's Self-Organising Map (SOM) [4], [5] , and (ii) second using the frequencies of n-gram occurences.

In the first part of this work, we employ the SOM to determine the semantic relevance of a "candidate" translated term with respect to its context, and thus, allow for a quantitative comparison among all the available alternatives that are suggested as candidate translations by a bilingual dictionary. In essence, using a monolingual corpus, we associate each encountered word from the corpus with a single matching from the SOM given its context. In practice, words that appear together frequently seem to be associated with neighboring matching units. As a result, we can compare different translations by adding the distances between all consecutive pairs of translated words and then selecting the translation that corresponds to the minimum overall route.

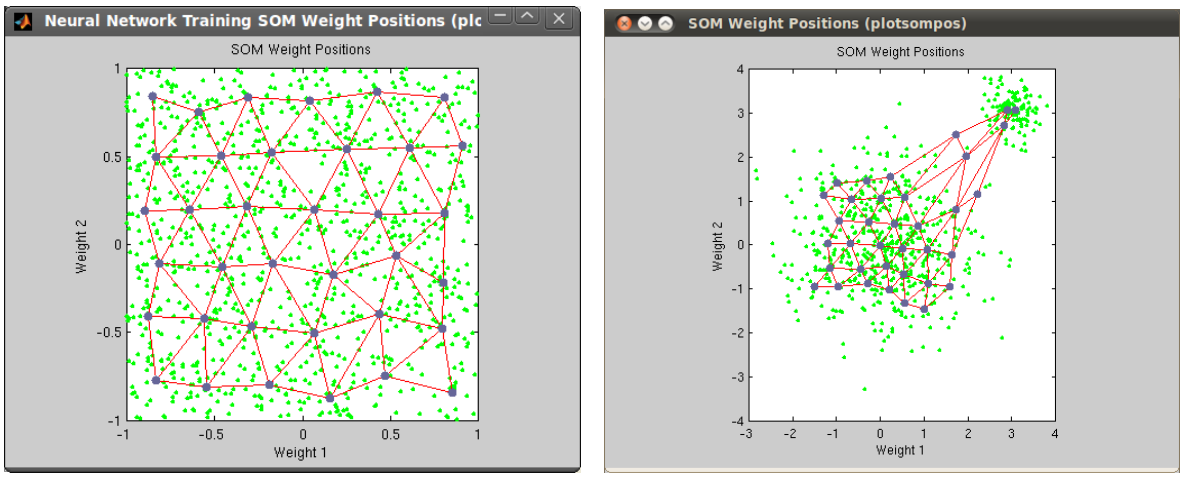
The second part of this work presents a conventional approach which is based on n-grams. An n-gram is a contiguous sequence of n words from a given corpus. An n-gram model is a type of statistical language model for predicting the next item in such a sequence in the form of a $(n-1)$-order Markov model. Hence, we are able to quantify how probable is a single combination of m translated words by forming $m-n+1$ n-grams and then multiplying the probability of all separate n-grams, as if they were independent from each other. Therefore, given all possible translations of a single sentence we are able to select the most likely to appear in the target language. Finally, additional models are developed using the aforementioned approaches, and we report results demonstrating their effectiveness.

2 SOM-based Disambiguation

For the purposes of this project with respect to word-translation disambiguation [9], a specialized language model is created by using the Self-Organizing Map (SOM) [4],[5]. SOM is a type of artificial neural network that is trained using unsupervised learning

to produce a two-dimensional, discretized representation of a high-dimensional input space of the training samples. Self-organizing maps are different from other artificial neural networks in the sense that they use a neighborhood function that changes over time in order to preserve the topological properties of the input space. More specifically, self-organizing maps learn to classify input vectors according to their similarity in the patterns space. Thus, self-organizing maps learn both the distribution and topological relation of the input vectors they are trained on and map the pattern space onto the output layer neurons. During training, the neuron that is located closest to an input vector is selected to adjust its weight vector toward those input vectors. Specifically, the network first identifies the winning neuron (or alternatively Best Matching Unit or BMU) for each input vector. Then, each weight vector moves to the average position of all of the input vectors for which it is a winner or for which it is in the neighborhood of a winner. The distance that defines the size of the neighborhood is altered during training through two phases, the first corresponding to the rough training and the second to the fine-tuning step. During rough training, the input patterns are ordered relative to one another while in the fine-tuning phase the weight vector of each node is fine-tuned to specific patterns. The neurons in the SOM output layer are arranged in a lattice with either square or hexagonal topologies. In the present work, a hexagonal topology is used. Figure 1 depicts a two-dimensional example of how the BMUs of a hexagonal topology are arranged into space for a uniform and a clustered dataset (two gaussian clusters). The popularity of SOM in terms of diverse applications is due to its flexibility and efficiency in unsupervised clustering tasks. The novelty of the SOM application in the PRESEMT prototype focuses on its integration in an MT system for word translation disambiguation. In the context of language processing, the features chosen to map the linguistic data to SOM are frequencies of occurrence of words within the sentences.

In the chosen approach, the input set consists of multi-dimensional vectors that describe the co-occurrences of encountered lemmas with a well-defined class of representative words. What makes this approach particularly attractive in the context of PRESEMT is that, in order to model a monolingual corpus, it does not require any external knowledge resources besides a large text corpus, the modeling process is fully unsupervised in the creation of the map and most of the processing is performed off-line. During the actual machine translation process, only the final SOM-generated mapping of words onto the map lattice needs to be accessed for disambiguation. This mapping is compact in terms of memory required and thus can be processed very quickly and



(a) uniform (b) clustered

Figure 1: BMUs of a hexagonal SOM topology for two-dimensional datasets.

efficiently. Moreover, this work focuses on the definition of an effective method for determining the semantic relevance of a "candidate" translated term with respect to its context. Thus, a quantitative comparison and ranking is performed among all the available alternatives that are congregated in a set of lemmas. Towards this end, the monolingual corpora produced by the PRESEMT corpus creation and annotation module are used to train SOM maps. In particular, our training set consists of multi-dimensional vectors that describe the co-occurrences of encountered lemmas with a set of feature words, determined using frequency-based criteria to exclude both very frequent (such as function words) and very rare words, via a modification of the ABC analysis. Once this language model is available, an adaptation of the well-known Viterbi algorithm is used for the Translation equivalent selection module in order to empower the overall optimal phrase selection efficiently. This task consists in selecting one lemma from each set and that way disambiguating multiple translations of single- or multi-words units. To elaborate, for the i-th alternative term of the phrase in the target language, we compute the transition cost from all the possible previous word forms. We also consider recursively the cost of selecting those forms given previous transitions. Then, from all j different word forms that lead to the i-th term at the k-th position of the phrase, we set $cost(k, i)$ equal to $\min_j distance(n(k, i), n(k-1, j)) + cost(k-1, j)$, where the distance signifies the Euclidean distance of the winner SOM neurons for the corresponding terms. The optimal path reaching term i at position k contains the optimal sub-path reaching j, and thus, when selecting the next alternative word-form there is no need to expand and compute any suboptimal paths from j as they have been pruned before-

hand, thus reducing processing time. For example, in the trellis diagram of Figure 2, only the best paths that lead to the j translations of the $(k-1)$-th word of the sentence are kept. Then, we compute the cost of the shortest transition for each different translation of the k-th word and the route that corresponds to the minimum cost route plus the optimal transition is chosen.

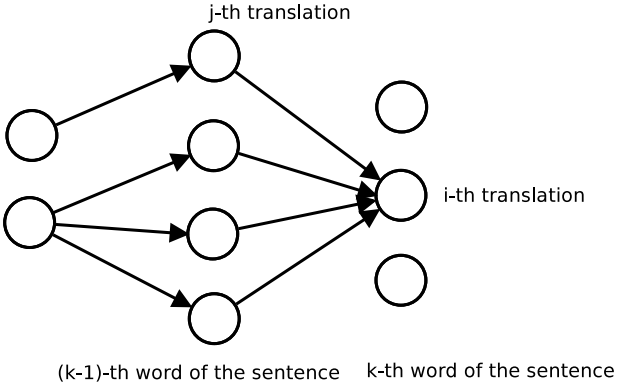


Figure 2: Trellis diagram example.

2.1 Implementation Details

In this section, we present the efforts we made to create an effective language map leveraging Self-Organizing Maps (also known as Kohonen maps). Apart from the disambiguation method for all tokens based on SOMs described earlier, we also developed a special version used to disambiguate first the heads and the functional heads of all phrases in a sentence, and then the rest of each phrase in isolation, having its heads fixed from the previous stage. To elaborate, we propose a two stage disambiguation paradigm. First, we congregate all words that are marked as heads and functional heads of the same sentence by themselves in the same structure with the same turn they appear in the sentence. As a result, for any given map, we evaluate the performance of two distinct methods, first the full-scale disambiguation noted as 'sentence-level', and the two-stage variation, noted as "FHP", that relies on head words. Then, in order to evaluate the disambiguation results from our system, we make use of a simplistic unigram-based measure, according to which we divide the number of disambiguated words which also appear in the reference translation (given by linguists and expert users) by the total number of assigned words, and then, we average the scores from all evaluated sentences. More formally, we have that,

$$\text{Score}_{ABCD|ABXY} = \frac{\#(\text{DisambiguatedWords} \cap \text{ReferencedWords})}{\#\text{AllWords}} \quad (1)$$

For example a disambiguated phrase containing words "A B C D", given the reference phrase "A B X Y", achieves a score of 50%, in a sense that half of the resolved words can also be found in the reference phrase.

Furthermore, for each SOM we also provide a 3-dimensional illustration which corresponds to a histogram indicating the distribution of words on the SOM. Specifically, for each neuron we draw a bar with a height analogous to the number of words for which the examined neuron is the best-matching unit.

2.2 ABC Analysis

ABC analysis was first incorporated into SOMs in [12]. The basic assumption made in [12] and also in this work is that words that occur frequently in similar contexts in natural-language expressions will bear related meanings. More specifically, the contexts considered here are sentences, i.e., text windows from one full stop to the next one. The use of such text windows is based on the hypothesis that the full stop between sentences is the least ambiguous point at which the description of an idea is completed. This basic hypothesis is frequently made in experiments involving word clustering. Ideally, for each lemma, the co-occurrences with all other lemmas would be recorded, although at the cost of a high feature-vector dimensionality. To limit this dimensionality, only a subset of available lemmas was chosen as the feature set, so that every lemma would be described by its cooccurrences with the lemmas from the feature set.

Pareto's principle, also known as the 80-20 rule [2] states that 20% of the causes are responsible for 80% of results. Pareto's principle, which is used in the ABC analysis, has mostly been applied to quality control and management tasks. According to the ABC analysis, a portion of the causes is characterized as A, which indicates important events, with B and C corresponding to less important and to unimportant events, respectively. In the word disambiguation application, category A contains highly frequent lemmas (corresponding to stop-words, such as articles, conjunctions, and auxiliary verbs, as well as other frequent words), B contains relatively frequent lemmas, and C contains rare lemmas. Lemmas from category B are selected for the feature set, since these lemmas do not correspond to very common words (that do not reflect a specialized content) yet are frequent enough to collectively describe all remaining lemmas. Initial limits of

the ABC analysis are set to implement an appropriate split of the input data in terms of frequency. For instance, in document organization applications on the basis of content [12], the following categories were used.

- Category A contains the most-frequent lemmas that collectively amount to a certain percentage (usually to 70%) of all occurrences.
- Category B contains lemmas that contribute the next 10%-15% of all occurrences.
- Category C contains lemmas that correspond to the remaining percentage of occurrences. In addition, to avoid studying exceptionally rare tokens, lemmas that occur less than three times throughout the corpus are omitted from category C.

B category is employed to represent each word from the corpus by its cooccurrences with the lemmas from the B category in a numeric vector for each word. More specifically, each lemma from categories A and C is represented by a vector of m elements, each indicating the number of times the given lemma cooccurs with the corresponding lemma from the B category. In order to implement the ABC analysis, initially each lemma's occurrences in the document set are counted. Then, the lemmas are ranked in descending order of frequency. Then category A is created iteratively, by introducing the current most-frequent lemmas in category A without substitution until the sum of normalized frequencies reaches the threshold of category A. When the sum ranges between thresholds A and B, the corresponding lemmas are assigned to category B and the rest are assigned to category C.

2.3 English Monolingual Language Models

According to the corpus used for training SOM, we separate the maps into four main sets, C_a, C_b, C_c and C_d. Set C_a SOMs correspond to all maps that were trained using a small English (filtered) corpus of 10 MBs in VERT format (three columns). Set C_b SOMs correspond to a larger corpus of 25 MBs in VERT format, set C_c to a 100 MBs corpus and set C_d to a large 1.5 GBs corpus. In corpus C_a 11050 distinct lemmas appear, set C_b corpus contains 18408 distinct lemmas, set C_c 42954. Last, set C_d corpus comprises 88090 distinct lemmas. Generally, the suggested dimensionality of a map is computed the same way it is computed in the SOM_PAK (`http://www.cis.hut.fi/research/som-research/`). When map dimensionality is manually set is made explicit in the experiment definition.

2.3.1 Standard Model

For the first SOM, we make use of the set C_a corpus whist the B category limits are set to .70 and .85 for the lower and the upper bound, respectively. There are three variations of this experiment and their only difference involves the number of cycles allowed during training. The size of the SOM was automatically selected, based on a SOM criterion, and set to 14 by 36 at all times. In addition, three-dimensional diagrams of these maps depicting histogram frequencies are shown in Figures 3 and 4. Moreover, Figures 5, 6, 7, 8 and 9 depict all the intermediate states SOM passes through until it converges to its final state in Figure 10. In Table 1 we present an overview of how scores scale as SOM training evolves with the number of iterations reached. Obviously, there is a remarkable increase in the quality of the results as the training process progresses. This is also indicated in the SOM evolution graphs in Figures 5-10. Apparently, the distribution of words to neurons becomes smoother as more iterations are performed. Especially, all major peaks that appear for the first 100 iterations are substantially reduced as many words are dispersed to neighboring neurons. To sum up with, we have three different configurations for our first set C_a map:

1. Rough stage training took 100 iterations while fine tuning took 20 iterations.
2. Rough stage training took 500 iterations while fine tuning took 100 iterations.
3. Rough stage training took 1000 iterations while fine tuning took 200 iterations.

Iterations	0	100	400	800	1000	1200
sentence-level	20.92%	25.38%	40.01%	46.34%	49.73%	50.3%
FHP	16.85%	22.15%	29.41%	34.67%	41.51%	47.3%

Table 1: SOM evolution scores.

We observed that disambiguation performance improves as the number of iterations increases. In particular, as presented in Table 2, we take for the first case 49.73% and 36.51%, for sentence-level disambiguation and FHP, respectively, for the second we have 49.8% and 40.1% for each method, and for the last method we take 50.3% and 47.3%. Clearly, there is a significant improvement for the FHP technique especially, since it starts from a lower level, but it still fails to outperform sentence-level disambiguation.

This same configuration was used for a much larger English corpus of approximately 100 MBs in VERT format (set C_c corpus). Due to the size of the corpus, we allowed 400

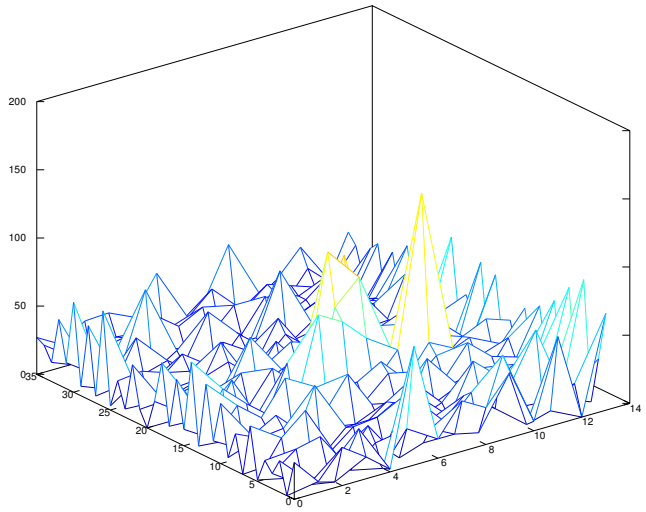


Figure 3: set C_a using B category bounds 70%-85% for 100 iterations.

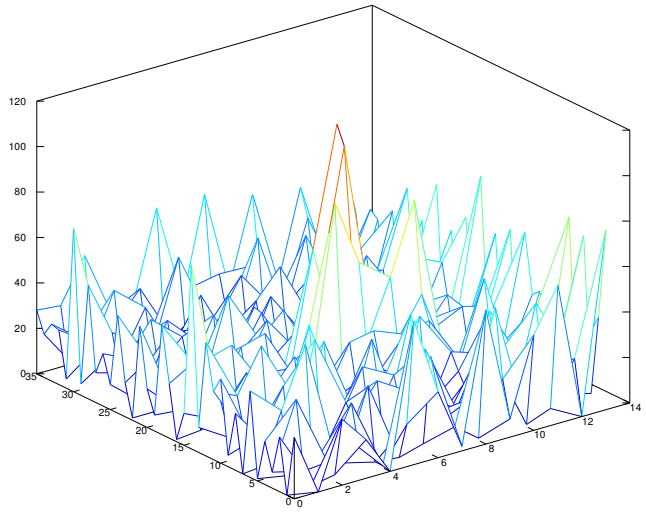


Figure 4: set C_a using B category bounds 70%-85% for 500 iterations.

Iterations	120	600	1200
sentence-level	49.73%	49.8%	50.3%
FHP	36.51%	40.1%	47.3%

Table 2: SOM evolution scores.

iterations and 80 cycles during fine tuning. The map's dimensionality was automatically selected and set to 17 x 56 . Even though we used a corpus ten times larger, the produced language model was not much better than the previous one. One can suspect that the limited number of iterations is responsible for the unexpectedly poor performance. In particular, the system produced a language model which when evaluated returned a score of 49.6% for the sentence-level disambiguation, and 46.7% for FHP. A three-dimensional representation of this is shown in Figure 11.

In Figure 12, we present a disambiguated example for a Greek sentence. We note that the referenced sentence was "British scientists believe that pomegranate juice may help us overcome stress at work". Thus, this specific result achieves a score of $\frac{8}{15} \approx 53.3\%$ (articles are not considered in calculating the score).

2.3.2 Increasing the Number of Training Iterations

The same small filtered English corpus (set C_a corpus) was used in another experiment. However, a great amount of iterations were used this time, as rough stage training took 5000 iterations while fine tuning took 1000 iterations. Map's dimensionality was automatically selected and set to 14 x 36 nodes. However, the system produced identical scores with our first experiment for 1000 iterations, 50.3% and 47.3%, for sentence-level disambiguation and FHP, respectively. The reason this attempt failed to produce better results was probably the limited size of the corpus. A three-dimensional representation of the distribution of words to SOM nodes is shown in Figure 13.

Therefore, for our next experiment we investigate the effect of training using a larger corpus (set C_b corpus), 25MB compared to the 10MB previously used, even at the cost of a reduced number of iterations, namely 1000. Indeed, introducing more information into the system results in an improved disambiguation performance. Specifically, we got 51.1% and 49.4%, for sentence-level disambiguation and FHP, respectively. A three-dimensional representation of this is shown in Figure 14.

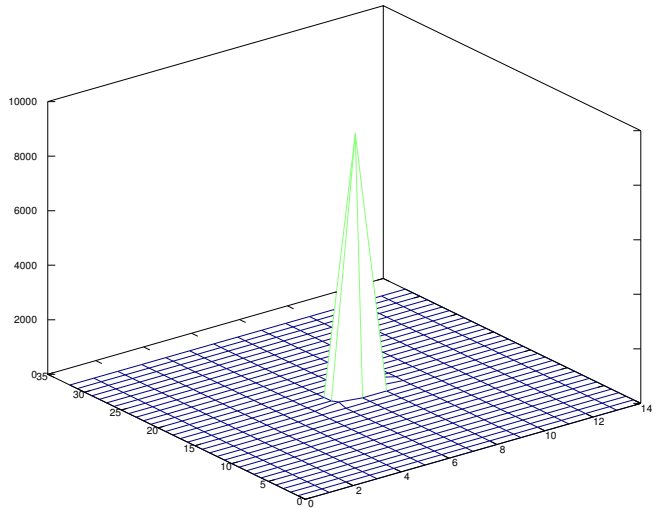


Figure 5: Representation of the training process for a set C_α map with B category bounds set to 70%-85% for 0 iterations (initialization).

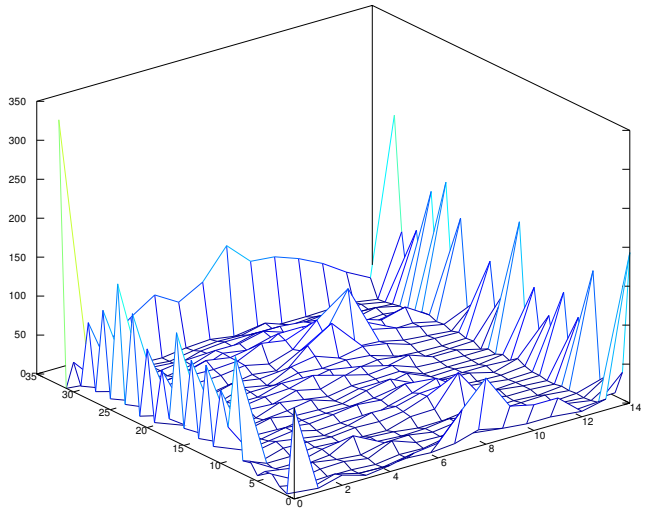


Figure 6: Representation of the training process for a set C_α map with B category bounds set to 70%-85% for 100 iterations.

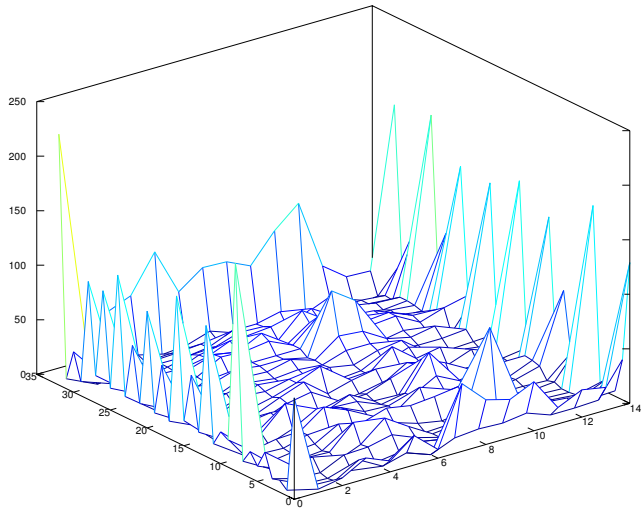


Figure 7: Representation of the training process for a set C_α map with B category bounds set to 70%-85% for 400 iterations.

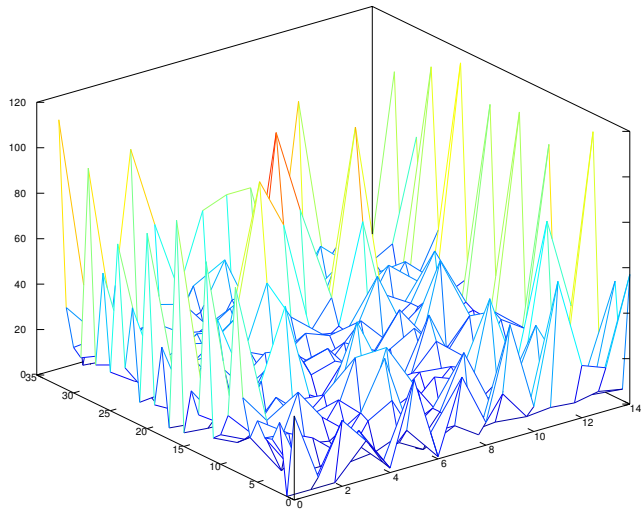


Figure 8: Representation of the training process for a set C_α map with B category bounds set to 70%-85% for 800 iterations.

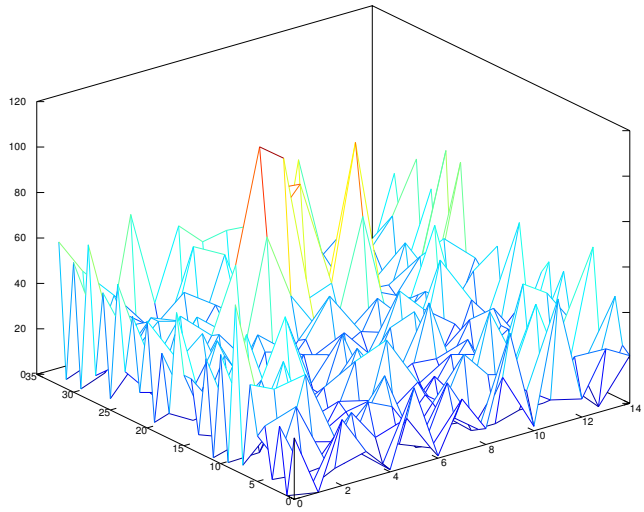


Figure 9: Representation of the training process for a set C_a map with B category bounds set to 70%-85% for 1000 iterations.

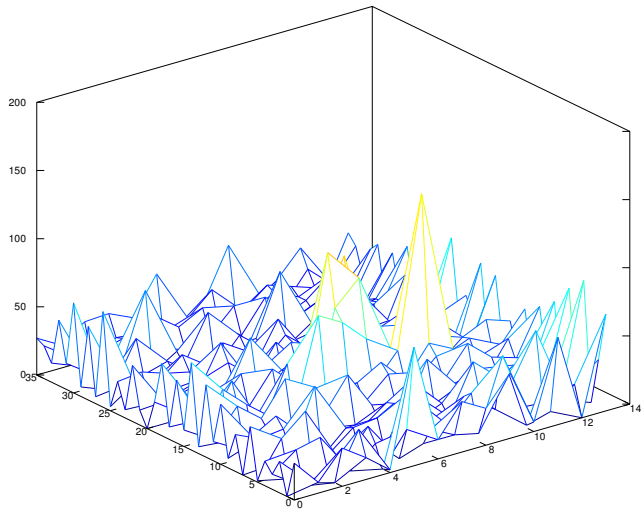


Figure 10: Representation of the training process for a set C_a map with B category bounds set to 70%-85% for 1200 iterations.

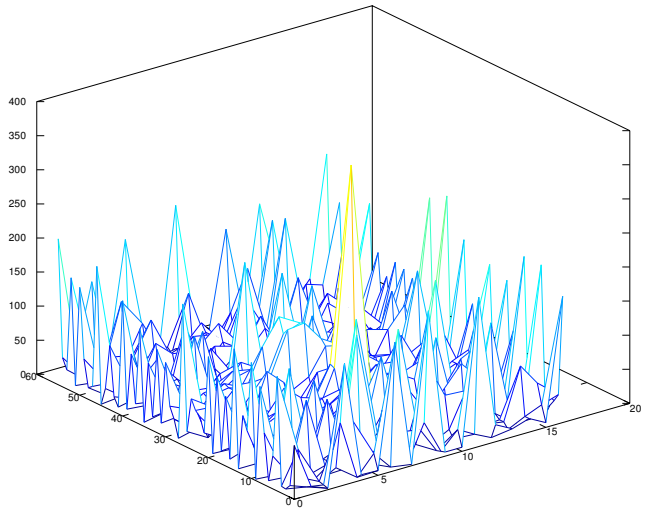


Figure 11: set C_c SOM using B category bounds 70%-85% for 480 iterations

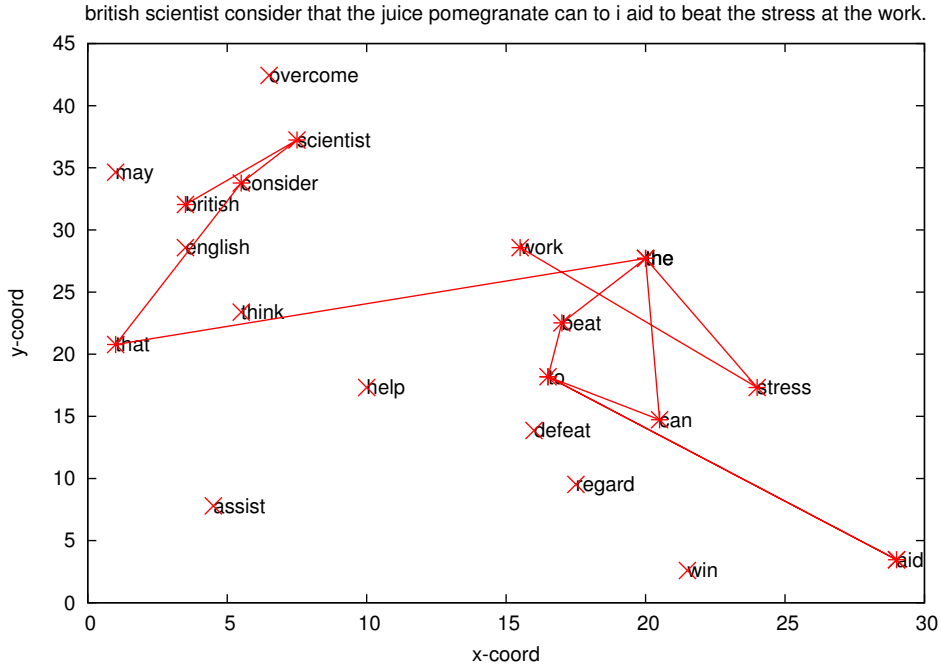


Figure 12: A representation of the disjunctive words over SOM along with the selected route that includes the selected words.

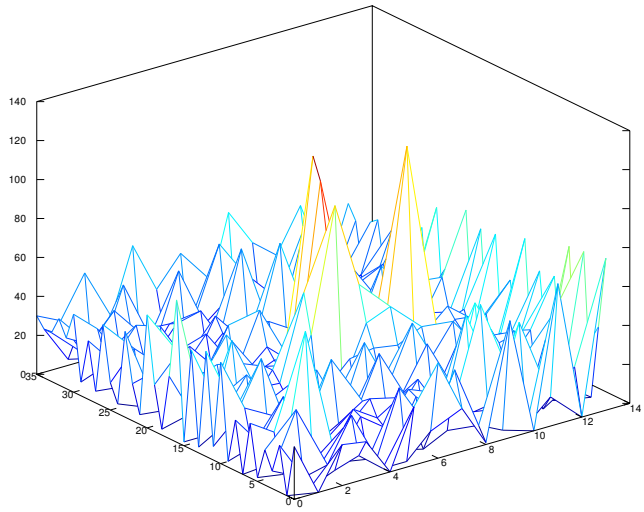


Figure 13: set C_a SOM using B category bounds 70%-85% for 5000 iterations.

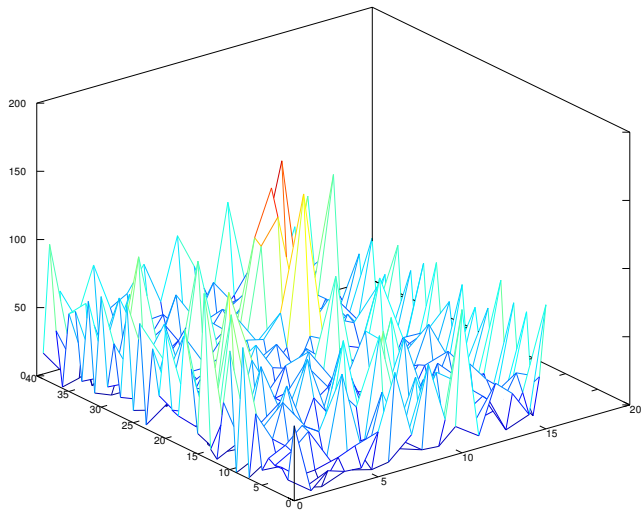


Figure 14: set C_b SOM using B category bounds 70%-85% for 1000 iterations.

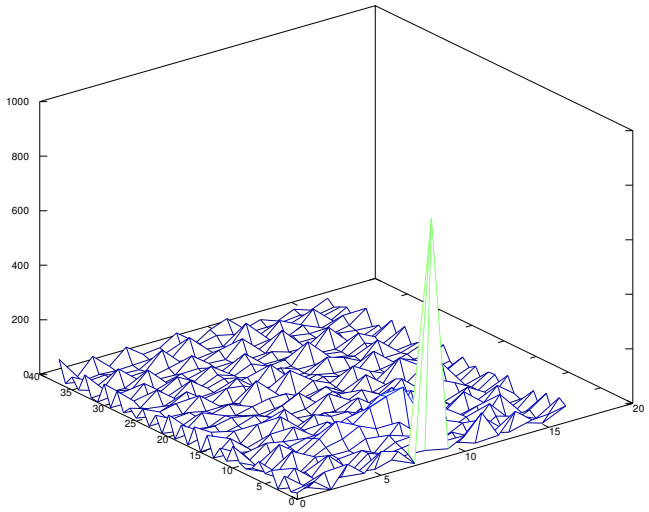


Figure 15: set C_b SOM using B category bounds 70%-85% for 5000 iterations.

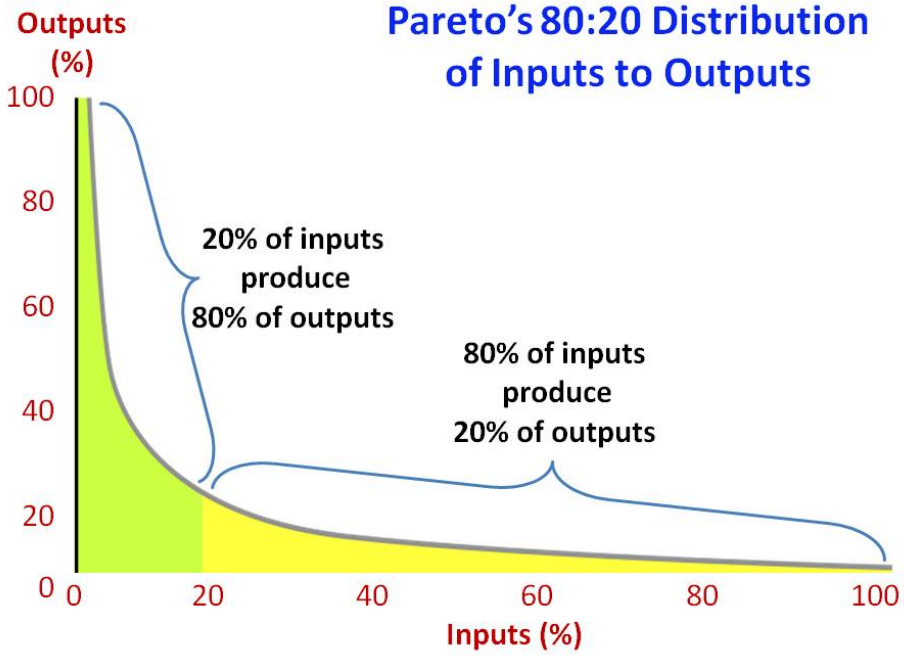


Figure 16: The Pareto cumulative distribution that the ABC analysis is based on.

2.3.3 Appropriate Configuration of B Category Bounds

The next experiments from set C_b are critical as certain conclusions will be drawn to allow for training using larger corpora. Since there is no chance of keeping the same limits for category B (.70-.85) due to the fact that extremely large feature vectors are produced for corpora over 100 MBs, we have to investigate how it is possible to reduce the breadth of category B, but still preserve as much information it is possible. Table 3 presents the number of features for different configurations for each of the used corpora. A great number of iterations was reached during training for both experiments: 5000 cycles for rough stage training and 1000 iterations for fine tuning. We also used the same corpus we used in the previous experiment. For the first instance of this experiment we set C_b category limits set to .70 and .75 for the lower and the upper bound, respectively, whereas for the second instance the same bounds are set to .80 and .85. We choose these bounds since they comprise words that are neither too rare, nor appear extremely often. In fact, the first approach was more successful than the latter, even though both windows share the same breadth (.05). Specifically, we got 51.1% and 48% for sentence-level disambiguation and FHP, respectively, from the first experiment, and 50.1% and 47% from the latter. This is due to two reasons. First, the first case contains more words. As shown in Figure 2.3.2, using the ABC analysis, the 70%-75% of outputs corresponds to fewer features than the 80%-85%. Second, the first group of words is a little further than the extremely frequent words from which we cannot extract enough information as they appear almost in every sentence, e.g. conjuctive words, prepositions, etc. Therefore, we henceforth set the limits of the B category to .70 and .75 whenever we use a large corpus which exceeds in size the 100 MBs threshold. The dimensions of the maps were 17 x 38 and 19 x 34, respectively. Three-dimensional representations of these maps are shown in Figures 15 and 17. The prominent peaks appear for neurons to which a disproportional amount of words are matched. In our case, having just a few overloaded neurons is a great problem since the ability to discern one word from another by their matching units and their eligibility in a given translated sentence is ineffective. Therefore, we cannot ascertain what translation to select from all the available translations returned from the dictionary that correspond to the same neuron.

	70%-75%	80%-85%	70%-85%
C_a	314	705	1448
C_b	347	848	1716
C_c	375	1001	2022
C_d	436	1262	2651

Table 3: Feature vector cardinality for different configuations.

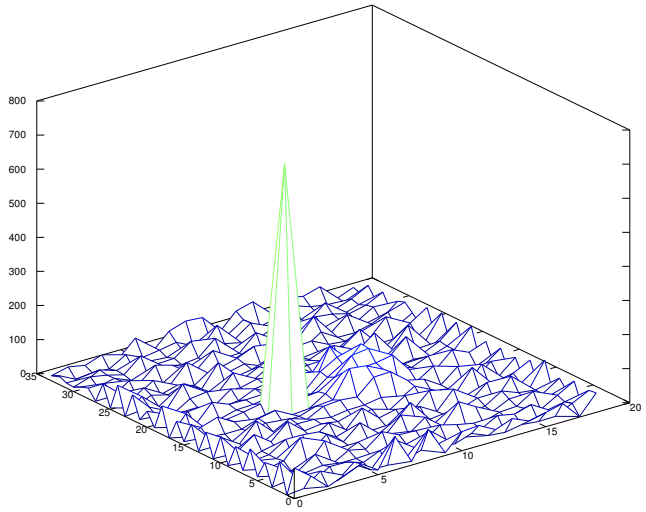


Figure 17: set C_b SOM using B category bounds 80%-85% and 5000 iterations.

2.3.4 The Effect of Map Dimensionality

For the next set C_a experiment, we wanted to study how the size of SOM affects disambiguation. We have already made an appropriate selection for what the breadth of a representative B category should be to allow for efficient processing of large corpora, while still carrying enough information in order to enable effective word translation disambiguation and achieve good performance. More specifically, using a small corpus of 10 MBs we created maps of dimensions varying from 10 x 20 for small maps, to 30 x 50. Indeed, there was a slight increase in the results of the small map (46.32% and 45.21%, for sentence-level disambiguation and FHP) to 50.28% and 48.4%, which can

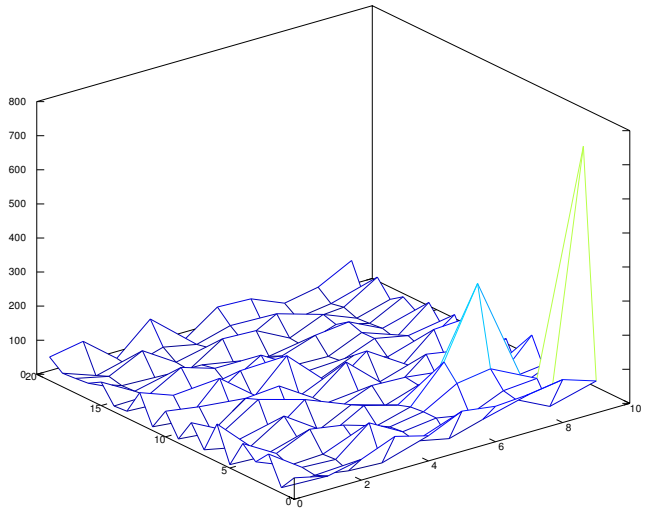


Figure 18: set C_b SOM using B category bounds 70%-75% and 2500 iterations.

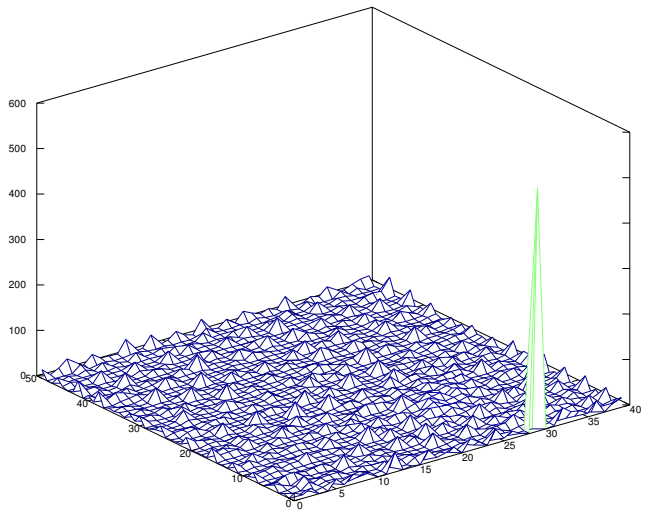


Figure 19: set C_a SOM using B category bounds 70%-75% and 3000 iterations.

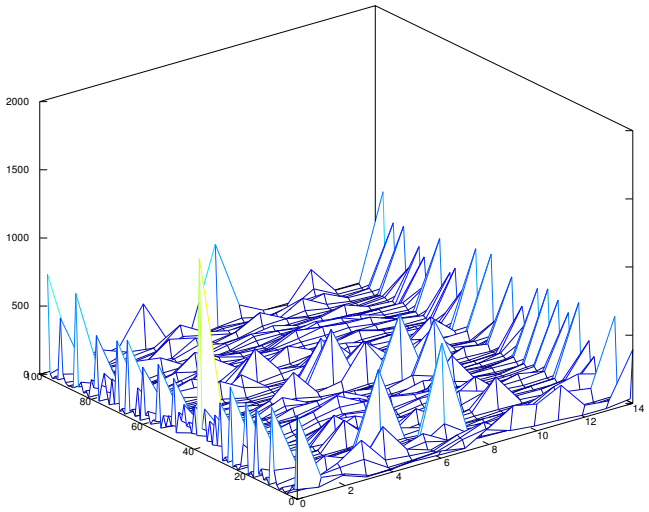


Figure 20: set C_d SOM using B category bounds 70%-75% and 1200 iterations.

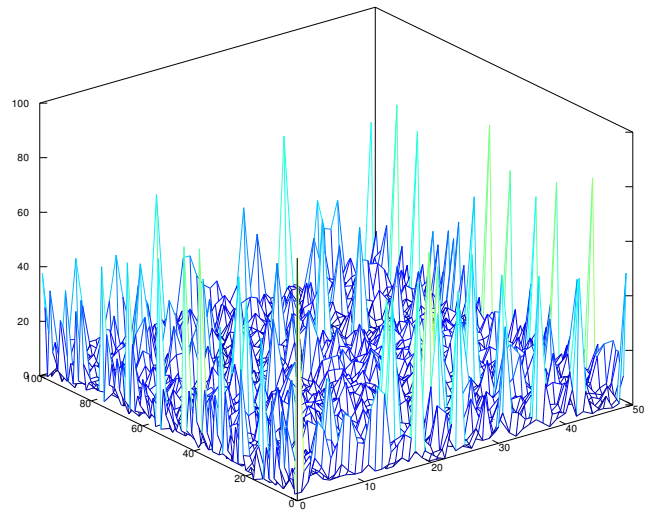


Figure 21: set C_d SOM using B category bounds 70%-75% and 2500 iterations.

further improve if we use a larger corpus. A three-dimensional representation of the SOM for C_b is shown in Figure 18, while the SOM for C_a is shown in Figure 19.

For the next SOM we trained, we used a corpus of approximate size 1.5 GBs in total in VERT format (set C_d). Map's dimensionality was automatically selected and set to 15 x 97. From Figure 20 we conclude that 1200 cycles in total (1000 iterations for rough stage training and 200 for fine tuning) are not sufficient to spread appropriately word vectors to the neurons of the map mainly due to the vast size of the corpus used. In other words, the more data you use to train a SOM, the more iterations you need. Specifically, the scores this map achieved were 51.3% and 48.6%, for sentence-level disambiguation and FHP, respectively. In fact when we repeated the same experiment for 2500 iterations in total (2000 for rough stage training and 500 for fine tuning) we achieved slightly better scores, 52.1% and 49.2%, for sentence-level disambiguation and FHP. The histogram of patterns assigned per SOM node of this map is shown in Figure 21.

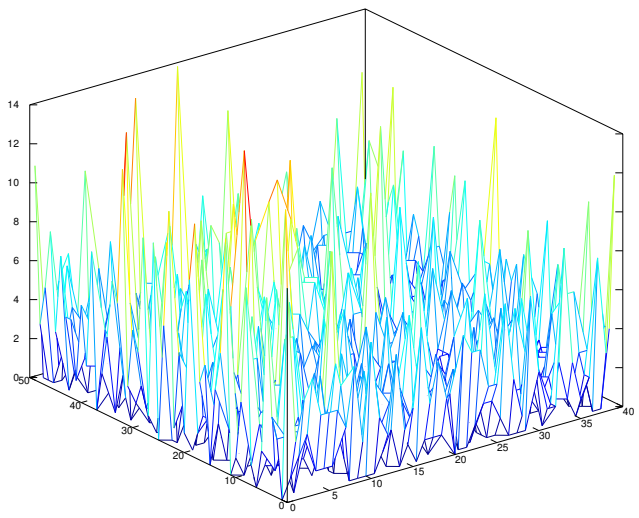


Figure 22: set C_a SOM using B category bounds 70%-75% and 3000 iterations with threshold set to 6.

In Figure 25 we compare our SOM disambiguation technique with other competitor schemes, like Vector Space Model (VSM) proposed in [8]. We depict with an orange dashed line a baseline disambiguation scheme that relies on the most frequent translation overall. With a red dashed line we depict another baseline which returns each time

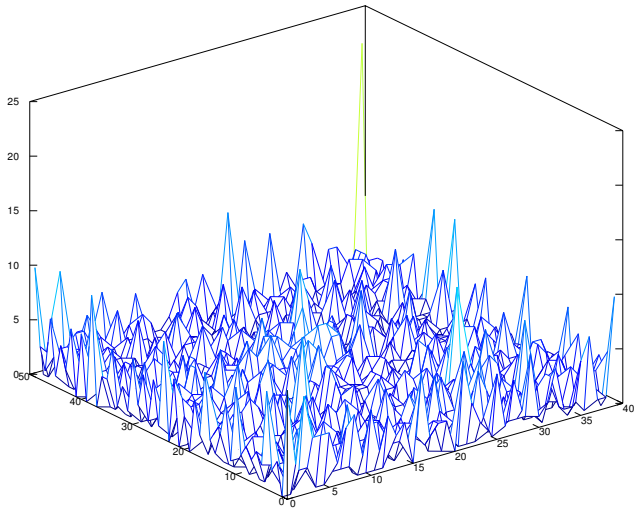


Figure 23: set C_a SOM using B category bounds 70%-75% and 3000 iterations with threshold set to 8.

the most frequent translation of the examined word in the source language. Yet, another approach which is based on five-grams is shown with gray. Quite remarkably, the red line which corresponds to the baseline which has been built on a parallel corpus is quite successful despite its simplicity.

2.3.5 Filtering by Tag

Our next step was to try a different approach to improve performance. In this set C_a experiment, we concentrate on removing from the training corpus the terms that do not impart the system with enough information to allow for effective disambiguation. These are instead considered to add noise to the system indirectly, and thus, overall performance deteriorates. Hence, we chose to remove from the training corpus all articles, prepositions, pronouns, possessive pronouns, predeterminers (all, half, etc.), wh-words (where, who, how, etc.), wh-pronouns (who, whom), adverbs, wh-adverbs, modal verbs, conjunctions and cardinals, etc. We also used the corpus of 25 MBs from which all words with frequency less that 10 were filtered out. We then manually set the dimensions of the resulting maps to 30 x 50. In addition, during training 2500 iterations

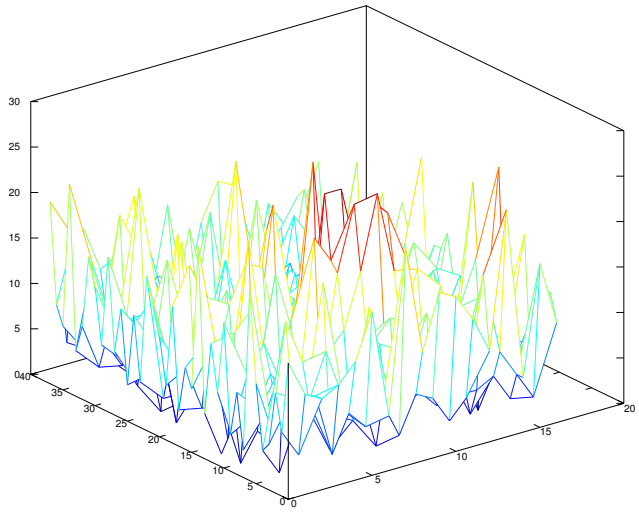


Figure 24: set C_α SOM using B category bounds 70%-75% and 3000 iterations with threshold set to 10.

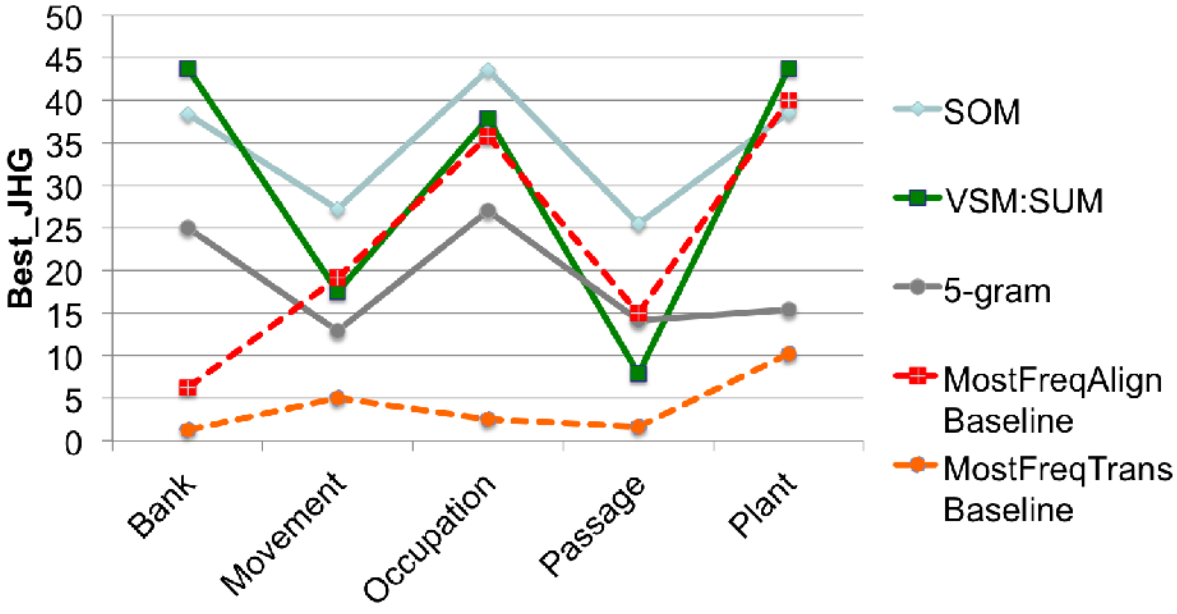


Figure 25: Comparison of different disambiguation methods.

were consumed, 2000 during rough stage, and 500 iterations during fine tuning. From the standard processed map, with this configuration, we received 50.21% and 47.56%

for sentence-level disambiguation and FHP, respectively. Then, after filtering out all the words that fall into the aforementioned categories, we created a language model from which the disambiguation results were 52.46% and 49.87% for sentence-level disambiguation and FHP, respectively.

2.3.6 Reduced Feature Vector

We recently augmented the SOM training process with certain optimizations to allow for the processing of larger corpora in less time. In particular, we concentrated our efforts on post-processing the extracted feature vector used to represent each word by a vector that corresponds to its co-occurrences in a sentence with a special category of words that carry the most information. Two aspects of this approach were investigated. (i) First, we defined a new parameter to specify the maximum length of the feature vector. If the length of the feature vector exceeds the declared parameter, say K, then only the K highest frequency constituents of the feature vector are kept, while the remaining elements are discarded. As a result, both memory usage and processing time are reduced due to the shorter vector used. (ii) We also implemented a modification of this technique according to which we repeat the same procedure for each word separately, allowing for a more precise intervention to the feature vector by keeping the K most frequent elements of the feature vector for each word. In addition, these two methods can be combined naturally.

This experiment aims at studying the effect of the threshold parameter which ranges from 2 to 10. The main reason we did not proceed with threshold values larger that 10, lies with the fact that we observed that normal words together the insignificant and the incorrect ones seem to be cut off. At all cases, map dimensionality was manually set to 40 x 50. In particular, we trained all these maps using the set C_a corpus and each experiment reached 2500 iterations for rough stage training and 500 iterations for fine tuning. In Table 4 we present analytically the results of each threshold we test.

Threshold	2	4	6	8
sentence-level	50.81%	50.93%	51.20%	51.55%
FHP	48.53%	48.41%	48.56%	49.2%

Table 4: Disambiguation results for varying cut-off frequency thresholds for the rarest words of the corpus.

In Table 5 we show the effects of reducing the feature vector for a set C_b experiment. A reduction of up to 20% ameliorates translation quality only slightly, though. However, for even greater reduction scores seem to diminish instead, as expected due to information loss. Finally, the gains in time and memory consumption were very small.

Reduction	Max Features	Score	Time/Iteration	Memory
0%	213	57.17	127″	862 MBs
20%	151	57.19	123″	824 MBs
30%	149	57.02	124″	823 MBs
40%	128	56.32	123″	819 MBs
50%	107	56.44	121″	815 MBs
60%	85	55.53	121″	814 MBs
70%	64	54.65	106″	812 MBs
80%	43	54.50	105″	750 MBs

Table 5: Results for the 25MB English corpus.

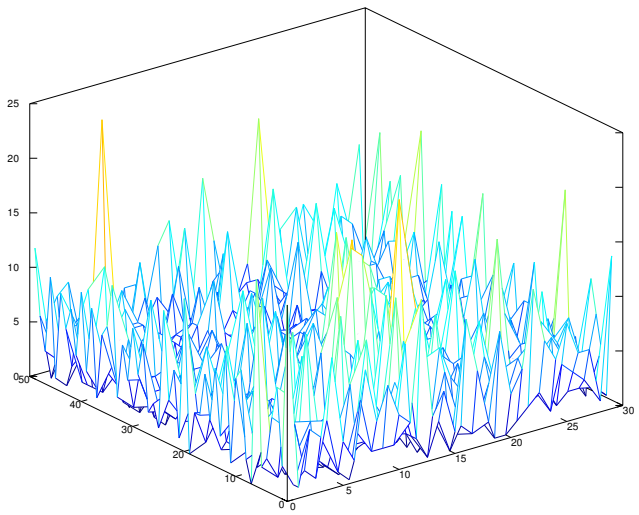


Figure 26: set C_b using B category bounds 70%-75% and 3000 iterations with dimensionality 15 x 20

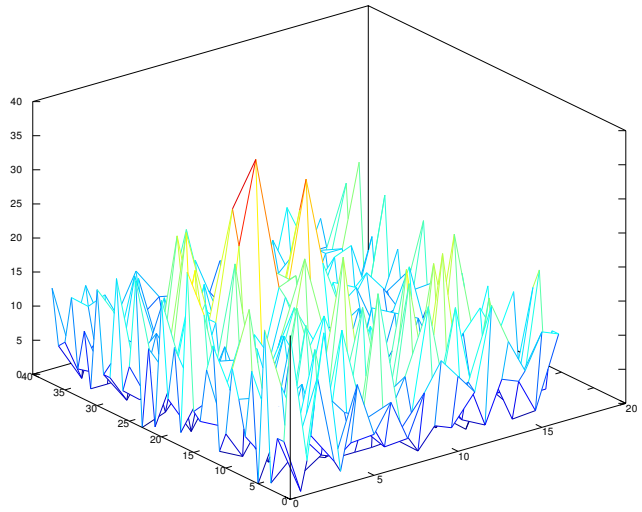


Figure 27: set C_b using B category bounds 70%-75% and 3000 iterations with dimensionality 17 x 38

2.4 German Monolingual Language Models

Following the same approach, we developed SOMs that were used for disambiguation for German as the target language. Such an example is illustrated in the map histograms in Figures 35 and 36. In particular, we used a German corpus of 100 MBs to train two SOMs with different limits for the the B category. Specifically, the map presented in Figure 35 uses during training a feature vector containing the words that correspond to the 80%-82% of the occurences in the corpus. Analogously, for the map shown in Figure 36 we set the bounds of the B category to 70% and 80%. Clearly, in the latter map we see a more uniform distribution which is more appropriate for the disambiguation task, as it groups words into small clusters in terms of their context in the corpus, and hence, we are able to distinguish a good choice from a bad one when asked to resolve a disjunction arising from the numerous translations of the lexicon from each word in the source language. On the other hand, the map shown in Figure 35 shows an uneven distribution of words into neurons which is rather inconvenient for our purposes since a large proportion of the vocabulary is associated with just a few neurons, and thus,

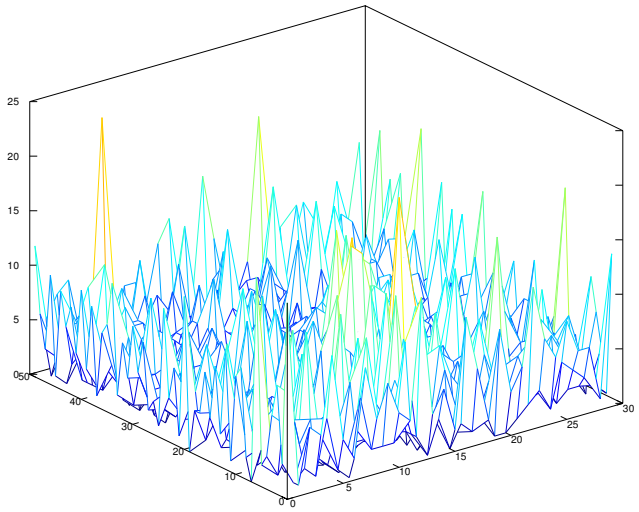


Figure 28: set C_b using B category bounds 70%-75% and 3000 iterations with dimensionality 30 x 50

makes it impossible in many cases to discern one word from another in terms of their matching units.

3 N-gram based Disambiguation

We have developed a module for the same purposes that relies on a language model which leverages n-gram modeling techniques. However, in our case an n-gram is a contiguous sequence of phrase heads and/or functional heads. More specifically, our language model consists of a composite model that combines trigrams and bigrams, as well. In particular, if a specific trigram, say (a, b, c), is absent from the language model that we have extracted from the available English corpus, then the specific trigram is analyzed into bigrams, namely (a, b) and (b, c). Next, assuming independence, we are able to compute the probability of the initial trigram $p(a, b, c)$ from the product of $p(a, b)$ and $p(b, c)$ using our complementary bigram language model. On the downside, no smoothing technique is used for the unseen words of n-grams.

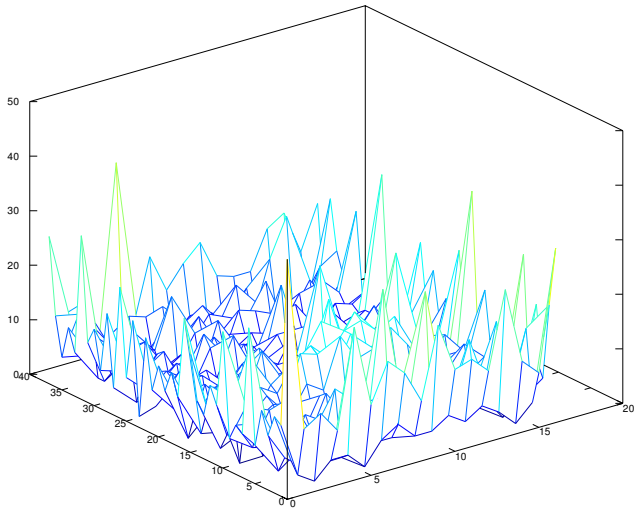


Figure 29: set C_b SOM using B category bounds 70%-75% and 3000 iterations with filtered 20% of the rarest words

The input of the algorithm consists of a sequence of disjunctive heads and/or functional heads $(d_1, \cdots, d_n)$. Each disjunctive word expresses a number of different word translations for the corresponding word in the source language. As a result, a large number of translations can be compiled by combining different translations from each disjunctive word. Therefore, our main concern is to retrieve efficiently the most probable sequence of words given a sequence of disjunctive words. In this work, we adapt the well-known Viterbi algorithm accordingly.

To elaborate, the probability of a given sequence of n words, say $(w_1, \cdots, w_n)$, $p(w_1, \cdots, w_n)$ under our independence assumption it is equal to $p(w_1, w_2, w_3) \times p(w_2, w_3, w_4) \times \cdots \times p(w_{n-2}, w_{n-1}, w_n) = \prod_{j=1}^{n-2} p(w_j, w_{j+1}, w_{j+2})$. To this end, we examine in turn all congregated words from each disjunctive word in the given sequence. Then, when the i-th word w_i from the m-th disjunctive word d_m is examined, the trigram probability $p(d_m.w_i, d_{m-1}.w_j, d_{m-2}.w_k)$ is retrieved from the extracted language model, which is stored on the disk, for each alternative combination of the previous words comprised in the disjunctive words preceding d_m. Note that the same combinations of words precede each word w_i comprised in d_m. However, each has a different occurrence probability generally when combined with w_i.

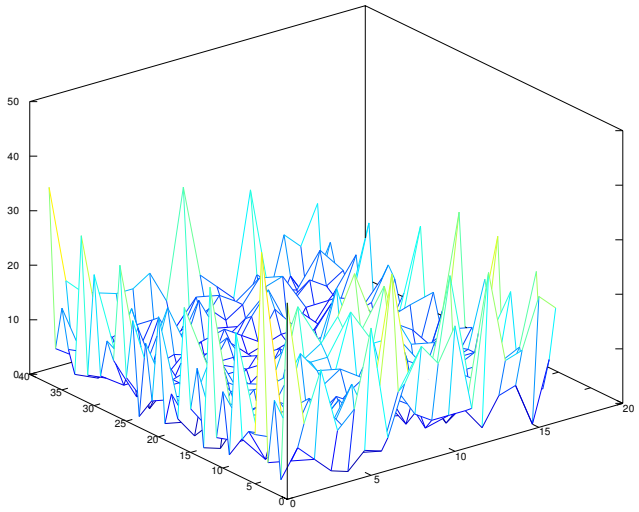


Figure 30: set C_b SOM using B category bounds 70%-75% and 3000 iterations with filtered 30% of the rarest words

In the core of the disambiguation process lies matrix P with length equal to the number of disjunctive words in the sequence. In turn, each element P_m has a length equal to the number of distinct word translations returned by the PRESEMT lexicon for the m-th word in the source language. Then, each element $P_{m,i,j}$, which is associated with word w_i from disjunctive word d_m and word w_j for disjunctive word d_{m-1} is set to the maximum probability of occurrence of the trigram $d_m.w_i, d_{m-1}.w_j, d_{m-2}.w_k$ from all k alternatives for disjunctive word d_{m-2}. Hence, $P_{m,i,j}$ will be set to $P_{m-1,j,k} \times p(w_i, w_j, w_k)$, if and only if, this is the combination with the maximum probability, whereas all other options are discarded, and thus, contributing to reduced memory requirements and processing time. Note that we also take into account the occurrence probability of the best previous sequence of words when we multiply the probability of the triple with $P_{m-1,j,k}$ which corresponds to the optimum combination of word translations up to the j-th word of the $m-1$-th disjunctive word $d_{m-1}.w_j$. Additionally, an auxiliary matrix J of the same dimensionality as P is used to store all options made for optimum subsequences of word translations. More formally, $J_{m,i,j} = \text{argmax}_x P_{m-1,j,x} \times p(w_i, w_j, w_x)$. After all disjunctive words have been examined, the path with the alternatives which altogether constitute the translation of maximum probability is formed by backtracking

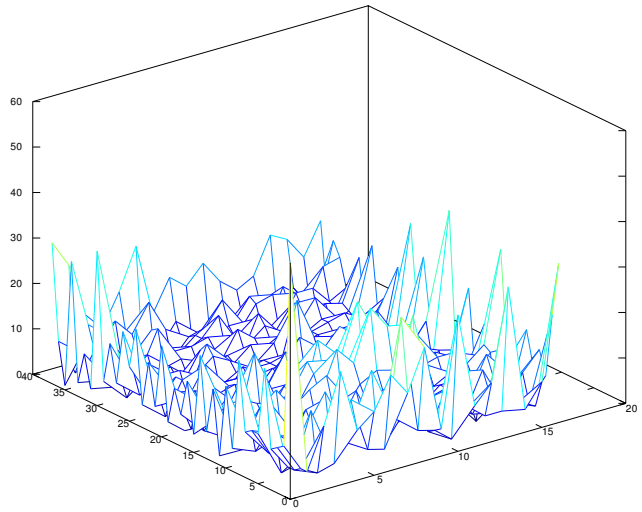


Figure 31: set C_b SOM using B category bounds 70%-75% and 3000 iterations with filtered 40% of the rarest words

and selecting the minimum "cost" translations. We start by finding the position of the last element of the path of the maximum probability $\text{argmax}_{x,y} P_{n,x,y}$, where n stands for the total length of the sequence of disjunctive words, and we then add the $J(n, x, y)$-th translation of the n-th (last) word translation to the result. Next, we proceed progressively with adding to the answer the next element at position $J(n-1, y, J(n, x, y))$, and so on ... The increased complexity of the algorithm emanates from the fact that instead of keeping track of just one transition for each possible word translation w_i, as we would do for bigrams, we need to process words and transitions in pairs as if they constitute one symbol, though, in such a manner that we are still able to tell them apart when needed.

Finally, results are shown in the next section where we combine this disambiguation method with other techniques in order to accomplish a desirable result. Towards this end, we use the SRILM toolkit [1] to process efficiently large monolingual corpora (of approximate size tens of gigabytes in our case) and build our language models.

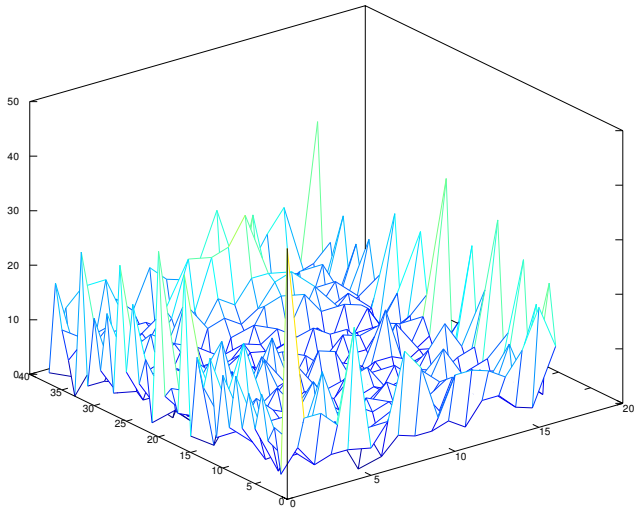


Figure 32: set C_b SOM using B category bounds 70%-75% and 3000 iterations with filtered 50% of the rarest words

4 Phrase Heads and Functional Heads Processing

Furthermore, we developed a special version for our SOM disambiguation tool and the n-gram disambiguation module, which are both used to disambiguate the heads and the functional heads of all phrases in a sentence. Specifically, we propose a two stage disambiguation paradigm. First, we congregate all heads and functional heads of the same sentence by themselves in the same structure with their initial turn. Next, all disjunctions are resolved using either of the aforementioned techniques, namely (i) SOMs, or (ii) n-grams, and the disambiguated words will replace the corresponding disjunctive words. Then, the rest of each phrase is processed separately according to a similarity criterion which compares each phrase with a large pool of similar phrase instances to select the best one (see [10] for more details on the technique). We also developed two main variations of this technique which ameliorate the quality of the results from the disambiguation process. The first variation of the original method includes extracting only the heads from the phrases producing a language model dedicated to disambiguating head words. Then a second separate language model is produced which is used for deciding prepositions, and functional head words in general, according to the already set head

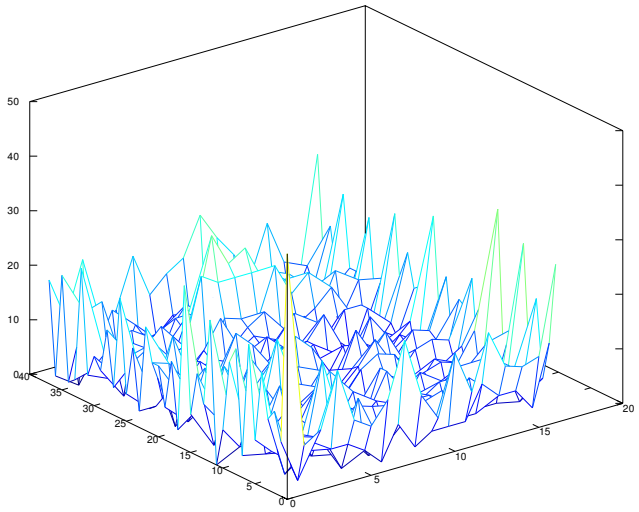


Figure 33: set C_b SOM using B category bounds 70%-75% and 3000 iterations with filtered 60% of the rarest words

words. Last but not least, we also created a language model for the disambiguation of verbs and nouns and then the remaining elements of each phrase are processed according to their frequencies in the monolingual corpus and the similarity of the phrase to already seen examples from the monolingual corpus.

In the following we present indicative examples for our head-processing paradigm. Consider the sentence: *"be the year {that/that/which/who} {drive/driven/guide/lead/result} {at/in/into/on/to/upon} {eruption/explosion/outbreak/outburst} of the civil war"* which is separated into five phrases, namely:

- *be*
- *the year {that/that/which/who}*
- *{drive/driven/guide/lead/result}*
- *{at/in/into/on/to/upon} {eruption/explosion/outbreak/outburst}*
- *of the civil war*

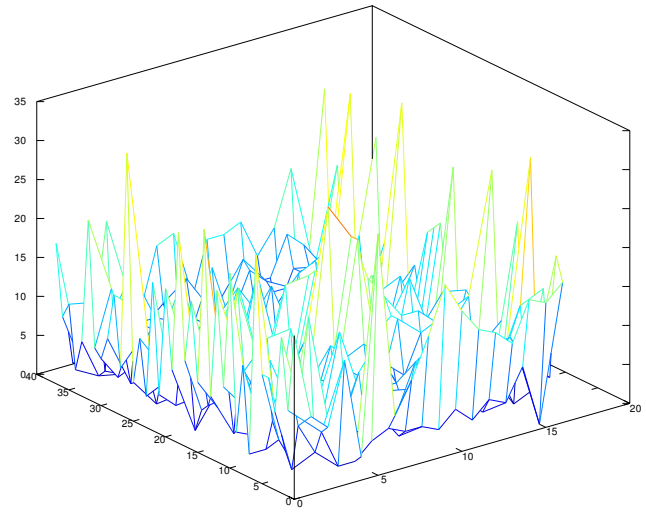


Figure 34: set C_b SOM using B category bounds 70%-75% and 3000 iterations with filtered 70% of the rarest words

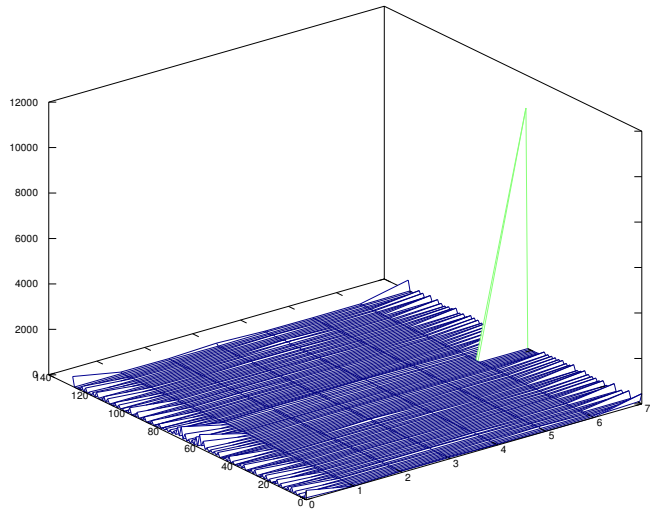


Figure 35: set C_c German SOM using B category bounds 80%-82% and 600 iterations.

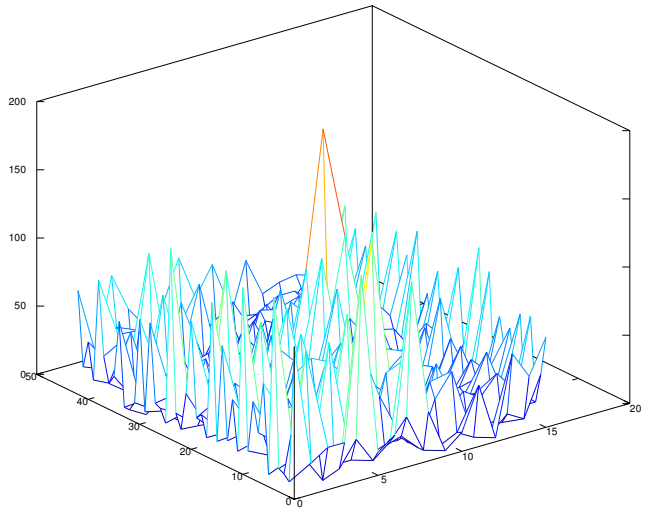


Figure 36: set C_c German SOM using B category bounds 70%-75% and 480 iterations.

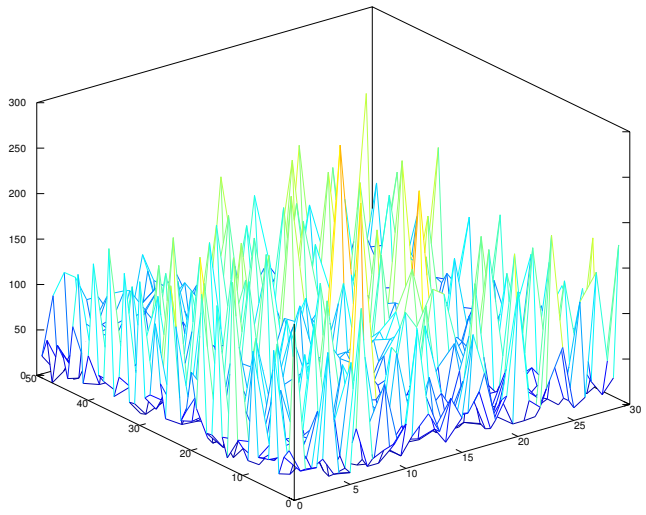


Figure 37: Training headSOM using English corpus (3GBs) with B category bounds 70%-85% and 1200 iterations.

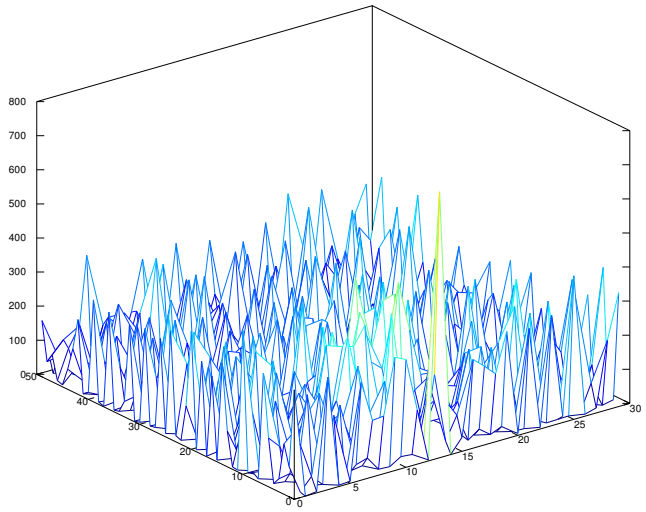


Figure 38: Training headSOM using English corpus (3GBs) with B category bounds 70%-75% and 1200 iterations.

Next, we extract the head and the functional head of each phrase in a single sequence of disjunctive and single words. More analytically, we have for each phrase:

- `head:` *be*
- `head:` *year*
- `head:` *{drive/driven/guide/lead/result}*
- `fhead:` *{at/in/into/on/to/upon}* `head:` *{eruption/explosion/outbreak/outburst}*
- `head:` *war*

Then, using our n-gram based disambiguator we end up with the following sequence of disambiguated head words and functional heads: *"be"*, *"year"*, *"lead"*, *"to"*, *"outbreak"*, *"war"*. On the other hand, using our SOM-based disambiguation paradigm we get: *"be"*, *"year"*, *"result"*, *"to"*, *"eruption"*, *"war"*. Given the referenced translation to be compared *"it be the year that lead to the outbreak of the civil war"*, we can easily observe that it coincides with the result from the n-gram based disambiguation example.

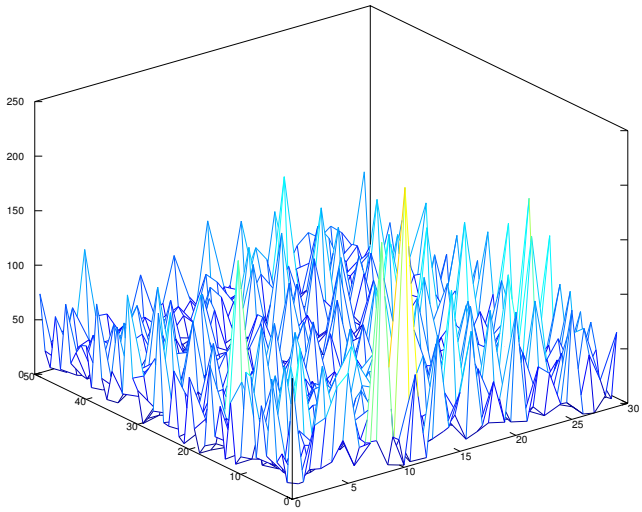


Figure 39: Training headSOM using German corpus with B category bounds 70%-75% and 500 iterations.

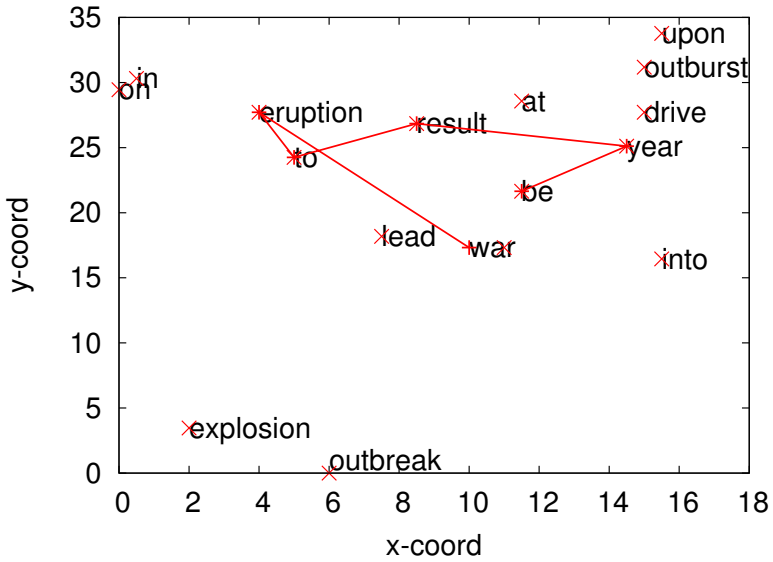


Figure 40: A representation of the disjunctive words over the Kohonen map along with the selected route that includes the selected head-words.

Overall, n-gram based disambiguation results are significantly better than the corresponding SOM-based disambiguation results. More specifically, we make use of an

indicative unigram-based measure which captures the success of a specific translation. To elaborate, for each sentence we divide the number of disambiguated words which also appear in the referenced translation by the total number of words. Then, we average the scores from all evaluated sentences, as in Eq. 1. Therewith, for a particular test-set consisting of 40 random sentences extracted from various sources, like press, blogs, web-sites, etc., we see that our SOM-based disambiguation paradigm achieves a general score of 45.61%, accounting for head and functional words only and not all possibilities, whereas the n-gram based disambiguation raises the bar a great deal by easily overpassing the 60% threshold, thereby, resulting in a performance gap of 18 units approximately.

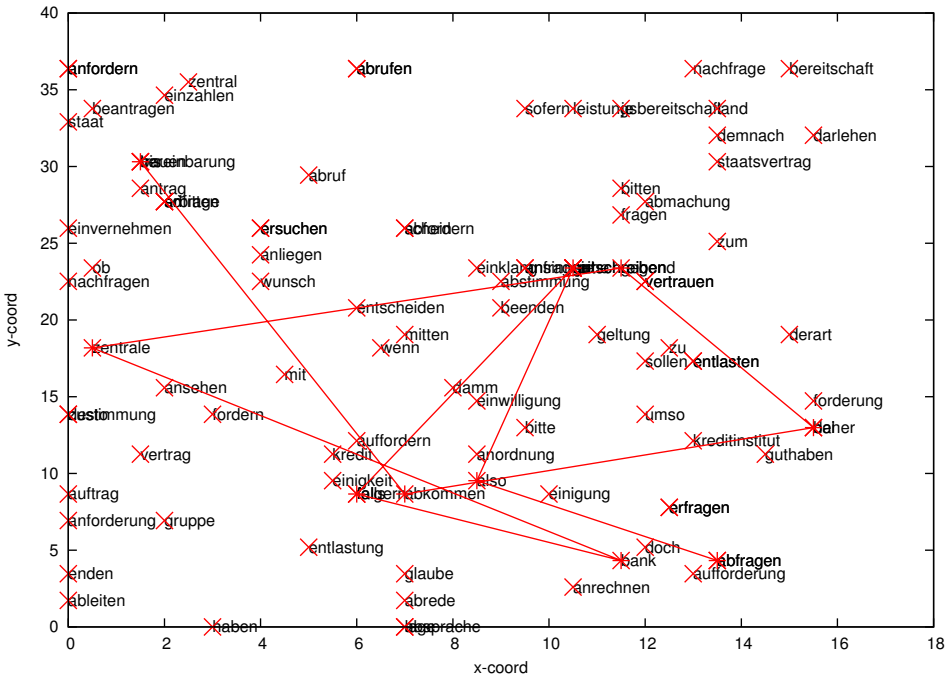


Figure 41: A representation of the disjunctive words over the Kohonen map along with the selected route.

Method	heads-fheads	j-heads	vn-heads
BLEU score	0.2557	0.2989	0.3202

Table 6: Disambiguation results for various head-based models.

Two successful variations of the aforementioned model, we will henceforth refer to as *heads-fheads*, were also developed. First, we tried disambiguating only the head words of the phrases, just like we described earlier, and then in each phrase separately

	Bank	Movement	Occupation	Passage	Plant	AVG
n-grams	38.64	35.82	31.25	23.48	30.77	31.99
SOM	31.25	30.77	20.0	15.38	21.43	23.77
Baseline	2.49	3.91	13.45	4.58	11.70	7.23

Table 7: Disambiguation results for the SEMEVAL benchmark.

we disambiguate each functional head word according to the best matching bigram using the predetermined head word from the previous stage. Henceforth, we will refer to this disambiguation method as *j-heads*. Second, an alternative model was implemented according to which we disambiguate just the head words and the functional head words whose part-of-speech is also verbs or nouns. Again, the rest of each phrase is disambiguated according to similarity and frequency criteria as proposed in [10]. In particular, we compare the remaining part of each phrase with a large pool of similar phrase instances to select the best one in the translation. We will henceforth refer to this disambiguation method as *vn-heads*. More specifically, in Table 6 we present the results for a development test-set of 200 sentences after we created language models based on an English corpus of approximately 10.5 GBs (in VERT format). The first basic method achieved a BLEU score of 0.2557, the second 0.2989 and the last 0.3202. In particular, our language model for the first approach, *heads-fheads*, includes 8002227 trigrams and 4221878 bigrams. The second approach, *j-heads*, considers 4739597 trigrams, since it uses only a subset of first approach where all included words are head words (functional heads are omitted) and 5402581 bigrams. It also uses the bigrams from the first approach as an auxiliary language model when resolving disjunctions for functional heads by considering the already determined head words. The last language model, *vn-heads*, which comprises only verbs and nouns includes 4702581 trigrams and 3739597 bigrams.

To the best of our knowledge, the most conspicuous benchmark in analyzing the strengths and weaknesses of word sense disambiguation (WSD) applications is SEMEVAL [7]. This evaluation procedure involves translating a set of 100 sentences, and each time we concentrate on the translation of a specific word. Specifically, these 100 sentences are divided into five groups of twenty sentences comprised in each group. In each of the five groups we compare the translations of a specific English word which exists in all sentences of the group as it is returned by our system with the correspond-

ing referenced ones. Namely, the words representing each group are: *bank*, *movement*, *occupation*, *passage*, *plant*.

However, since we want to analyze the performance of just the first stage of the disambiguation process, which involves *only* the disambiguation of head and functional head words, using each of the aforementioned techniques, we have to remove from each group the sentences which have their test words neither as a head word, nor as a functional head. Otherwise, our results would be obscured by the complementary disambiguation method of the second stage of the procedure and we would not be able to make solid judgements about the performance of our SOM-based and n-gram based disambiguation methods. Next, after isolating these sentences from the original benchmark out of necessity, we have 15 sentences for the sentence-group represented by the word *bank*, 17 for the group represented by *movement*, 16 for the *occupation*-group, all 20 for the *passage* group but only 10 for the *plant*-group.

Then, we construct result Table 7 using the normalized variant proposed by Jabbari et al. in [3], here referred to as $\mathtt{Best}_{\mathrm{JHG}}$. For each sentence t_i, with $1 \leq i \leq N$, where N stand for the number of test items, let H_i denote the set of human translations. For each t_i there is a function $\mathtt{freq}_i$ returning the count of how many annotators chose it for each term in H_i (0 for all others) and a value $\mathtt{maxfreq}_i$ for the maximum count for any term in H_i . The pairing of H_i and $\mathtt{freq}_i$ constitutes a multiset representation of the human answer set. Let $|S_i|$ denote the multiset cardinality of S according to $\mathtt{freq}_i$, i.e., $\sum_{\alpha \in A_i} \mathtt{freq}_i(\alpha)$, the sum of all counts in S. The $\mathtt{Best}_{\mathrm{JHG}}$ measure is defined as follows:

$$\mathtt{Best}_{\mathrm{JHG}}(i) = \frac{\sum_{\alpha \in A_i} \mathtt{freq}_i(\alpha)}{\mathtt{maxfreq}_i \times |A_i|} \tag{2}$$

where A_i is the set of translations for test item i produced by the system. The optimal score of 100.0% is achieved by returning a single translation whose count is $\mathtt{maxfreq}_i$, with proportionally lesser credit given to answers in H_i with smaller counts.

The baselines from [6] are based on the output of the GIZA++ word alignments on the Europarl corpus and just returns the most frequent translation of a given word. We observe a similar pattern of results emerging from Table 7. In Figure 41 we present an example for translating into German the sentence from SEMEVAL (English to German): *"der BIS koennen abschliessen Alarmbereitschaft Akkreditiv abkommen bei der glaeubiger gegend zentrale Bank falls sie haette also abfragen"*.

5 Conclusions

To recapitulate, we studied the problem of Word Translation Disambiguation (WTD) in the context of two completely different approaches. First, we consider the applicability of SOMs in order to resolve disjunctions among different translations. A number of routes is formed on the SOM the vertexes of which correspond to the selection of a specific translation. Hence, among all possible routes we select the shortest in a sense that is comprises translations that are more relevant to each other as they cooccur more frequently in the corpus used to train the SOM. Second, we implemented a conventional disambiguation module which relies on a combination of trigrams and bigrams. In particular, whenever an encountered trigram is not available in our language model, its probability is given by the normalized product of the probabilities of the two consecutive bigrams it is reduced to. Both approaches were scrutinized and studied meticulously under a number of different scenarios. However, the n-gram based approach was more effective and produced better results.

References

[1] `http://www.speech.sri.com/projects/srilm/`. *SRILM - The SRI Language Modeling Toolkit* supported by the SRI Speech Technology and Research Laboratory since 1995.

[2] Cam, R. and L. Gilles (2002). *A new model of the pareto effect (80:20 rule) at the brand level*. In Proceedings ANZMAC Conference, Melbourne, Victoria, Australia, pp. 1431 – 1436.

[3] Sanaz Jabbari, Mark Hepple, and Louise Guthrie. (2010). *Evaluation metrics for the lexical substitution task*. In Proceedings of the 2010 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, pp. 289-292, Los Angeles, California, June. ACL.

[4] Kohonen, T. (1982). *Self-organized formation of topologically correct feature maps*. Biological cybernetics 43 (1), pp. 59 – 69.

[5] Kohonen, T. (1997). *Self-Organizing Maps (2nd ed.)*. Springer Series in Information Sciences. Heidelberg, Germany: Springer.

[6] Lefever, E. and V. Hoste (2010a). *Baselines trial data, semeval package*. Technical report, University College Ghent, Faculty of Translation Studies.

[7] Lefever, E. and V. Hoste (2010b, July). *Semeval-2010 task 3: Cross-lingual word sense disambiguation*. In Proceedings of the 5th International Workshop on Semantic Evaluation, Uppsala, Sweden, pp. 15 - 20. Association for Computational Linguistics.

[8] André Lynum, Erwin Marsi, Lars Bungum and Björn Gambäck (2012). *Disambiguating Word Translations with Target Language Models*. In Proceedings of the 15th International Conference Text, Speech, Dialogue (TSD 2012), Brno, Czech Republic, pp. 378 - 385: Springer.

[9] Navigli, R. (2009). *Word Sense Disambiguation: a survey*. ACM Computing Surveys 41 (2), pp. 1 - 69.

[10] S. Sofianopoulos, M. Vassiliou and G. Tambouratzis (2012). *Implementing a language-independent MT methodology*. In Proceedings of the First Workshop on Multilingual Modeling, held within the ACL-2012 Conference, Jeju, Republic of Korea, 13 July 2012, pp.1-10.

[11] George Tambouratzis, George Tsatsanifos, Ioannis Dologlou and Nikolaos Tsimboukakis (2012). *SOM-based Corpus Modeling for Disambiguation Purposes in MT*. In Hybrid Machine Translation Workshop of the 15th International Conference Text, Speech, Dialogue (TSD 2012), Brno, Czech Republic.

[12] Tsimboukakis, N. and G. Tambouratzis (2011). *Word-map systems for content-based document classification*. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 41 (5), pp. 662 - 673.