Foundations Of LST II
Dr. Yoad Winter
2008/2009 blok 2

W. Bouvy 3079171
E. Szekely 3330060
6.02.2009

PROJECT REPORT FOR PIG

Partial Implementation of GUIneSS

TABLE OF CONTENTS

1. Introduction

1.1. Aim

We aim to build a program that is reliable in detecting agreement errors in several kinds of German adjective phrases, and is also able to give a suggestion to use endings that are more likely to be correct. We try to use a completely data-driven method, using n-gram frequencies coming from internet queries. Our reason for choosing for an entirely corpus-based approach, where the corpus is theoretically the whole internet, is that like this we do not have to be concerned with the problem of severe data sparseness. We called the approach GUIneSS: a Grammar Upgrader with Internet Search Statistics. The program that is described in this paper is restricted on agreement issues concerning adjective phrases. It can serve as a pilot project for GUIneSS, and is called PIG: a Partial Implementation of GUIneSS.

1.2. Hypothesis

The assumption we base this approach on is the following:

> *"A grammatically correct phrase will in general appear significantly more often than the same phrase containing an error."*

1.3. Problem

To test the method of GUIneSS, in PIG, we choose a rather local agreement issue in German: the inflection of adjectives.

In a German adjective phrase, the adjective has not only a gender, number and case agreement with the noun but its inflection form also depends on the determiner that accompanies the noun. The determiner (or the absence of the determiner) decides on the choice between three different inflection types that are often called strong, weak and mixed inflection classes. The next table[1] shows an example with a masculine noun. The variables are marked red. For the full inflection chart and further examples, please refer to Appendix A.

Gender	masculine	*ein* weicher *Stoff*
	neuter	*ein* weiches *Kissen*
	feminine	*eine* weiche *Landung*
Number	singular	*der* weiche *Stoff*
	plural	*die* weichen *Stoffe*
Case	nominative	weicher *Stoff*
	accusative	*ohne* weichen *Stoff*
	dative	*mit* weichem *Stoff*
	genitive	*statt* weichen *Stoffes*

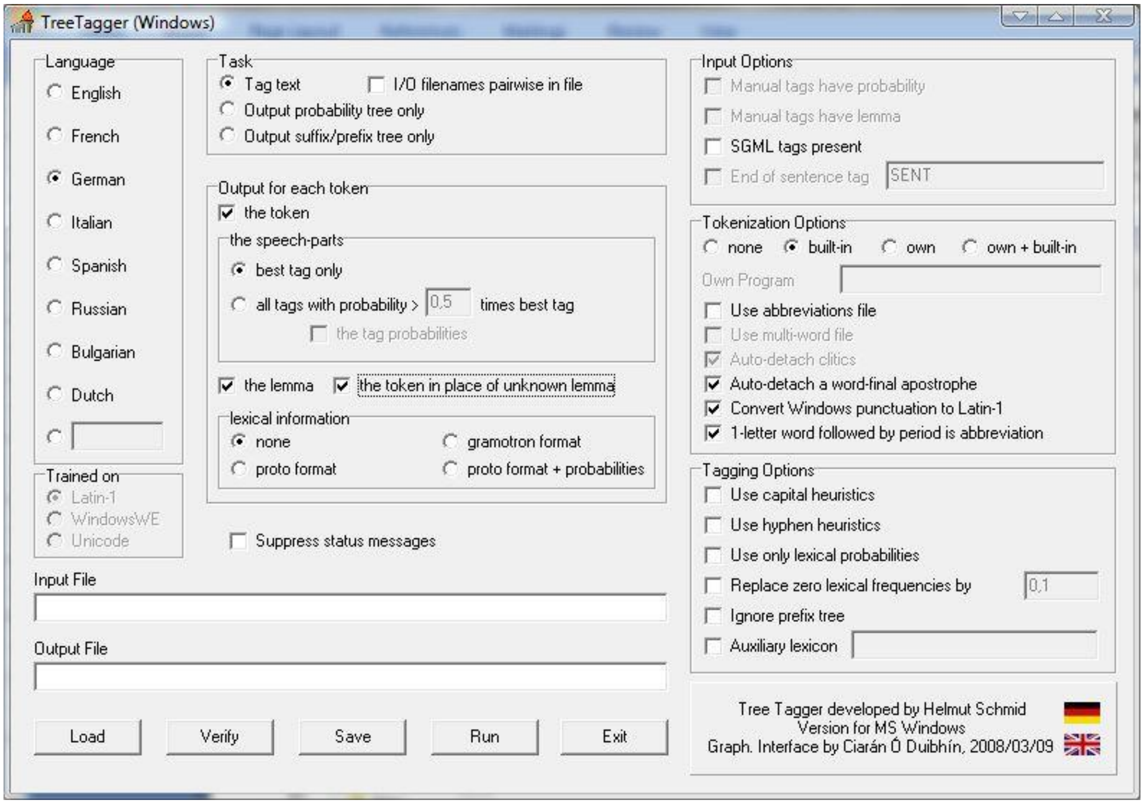
[1] the examples were taken from:
http://www.canoo.net/services/OnlineGrammar/Wort/Adjektiv/Deklinationstyp/index.html
accessed on: 27.01.2009

2. Data

2.1. The tagger

PIG makes use of an independent tagger application called TreeTagger. TreeTagger is an open source, language-independent part-of-speech tagger, developed at the Institute for Computational Linguistics of the University of Stuttgart. It can be used to annotate text with POS and lemma information. The text input is required to be tokenized before the tagger can be run.

Because we are using all online texts in the German language as a corpus, we obviously cannot run the tagger over our entire reference data. Instead, we use the tagger to annotate our input text and to create a lexicon with POS and lemma information that the program can refer to.



TreeTagger (Windows)

Language
○ English
○ French
● German
○ Italian
○ Spanish
○ Russian
○ Bulgarian
○ Dutch
○

Trained on
● Latin-1
○ WindowsWE
○ Unicode

Task
● Tag text □ I/O filenames pairwise in file
○ Output probability tree only
○ Output suffix/prefix tree only

Output for each token
☑ the token
the speech-parts
● best tag only
○ all tags with probability > 0,5 times best tag
□ the tag probabilities
☑ the lemma ☑ the token in place of unknown lemma
lexical information
● none ○ gramotron format
○ proto format ○ proto format + probabilities

□ Suppress status messages

Input File

Output File

Input Options
□ Manual tags have probability
□ Manual tags have lemma
□ SGML tags present
□ End of sentence tag SENT

Tokenization Options
○ none ● built-in ○ own ○ own + built-in
Own Program
□ Use abbreviations file
□ Use multi-word file
☑ Auto-detach clitics
☑ Auto-detach a word-final apostrophe
☑ Convert Windows punctuation to Latin-1
☑ 1-letter word followed by period is abbreviation

Tagging Options
□ Use capital heuristics
□ Use hyphen heuristics
□ Use only lexical probabilities
□ Replace zero lexical frequencies by 0,1
□ Ignore prefix tree
□ Auxiliary lexicon

Tree Tagger developed by Helmut Schmid
Version for MS Windows
Graph. Interface by Ciarán Ó Duibhín, 2008/03/09

2.2. The lexicon

Although we could have used the word list of the tagger itself, we decided against it because it would have had to undergo a serious editing process to fit our purposes. We needed a completely plain lexicon that contained every possible variation of a lemma. We therefore made use of the German spell check dictionary for PSPad that has been improved and corrected at the University of Duisburg-Essen[2]. It contains 592,430 entries. We run the tagger over this wordlist and created an annotated lexicon containing part-of-speech and lemma information. In cases where the tagger found the lemma unknown,

[2] downloaded from: http://www.uni-due.de/~gph120/woerterbuecher/
accessed on:16.12.2008

we made it repeat the same word form and set it as lemma. The next chart shows a sample from our lexicon:

TOKEN	POS	LEMMA
rotem	ADJA	rot
roten	ADJA	rot
röten	VVFIN	röten
roter	ADJA	rot
röter	ADJA	röter
roterde	NN	roterde
rotes	ADJA	rot
rötet	VVIMP	röten
rötete	VVFIN	röten
röteten	VVFIN	röten
rotfärbung	NN	Rotfärbung
rotfeder	NN	rotfeder

3. Algorithm

3.1. Overview

We begin with an outline of what the program does. In the following paragraphs we discuss the individual steps in detail.

After PIG reads a text from the input file or input window, it prepares it for the tagging by tokenizing it and runs the POS tagger on it. Then it extracts the phrases with the relevant POS combinations and generates a number of alternatives for the phrases, using all variations of the word that have a matching lemma. As a next step, it feeds each phrase into Google and runs several automated queries saving the numbers of hits. After having calculated percentages within every phrase set, PIG shows the results in a bar chart in a HTML interface.

3.2. Step 1: tokenization and tagging

The tagger will need an input file with every word on a separate line, so we split the input text on spaces and replace all punctuation with a full stop on a separate line. Commas are removed completely, as they can appear between adjectives that are still in one group. The resulting text is written to a temporary file for the TreeTagger. After this, the TreeTagger is called, so that the user can run it on the tokenized file. The TreeTagger window will appear, asking the user to give an input and output file. Afterwards, the user can resume the program by providing the output file of the TreeTagger.

3.3. Step 2: detecting n-grams

We designed POS combinations to detect phrases we are interested in the input text. We try to match any adjective phrase with or without determiner. We also match phrases with multiple adjectives. If there is a preposition prior to the adjective, we include it with the searches.

The program matches the combinations using a regular expression:

Regular expression:

```
(APPR)?(PPOSAT|ART|PDAT|PWAT|PIAT|PIS|CARD|PDS)?(ADJA)+(NN)
```

Explanation in words: If available, match a preposition; if available, match one of the determiner tags; match any number (1 or higher) of adjectives tags; and match the noun.

3.4. Step 3: generating alternative phrases

We get a set of phrases, for which we will generate alternatives.

The alternative phrases are supposed to represent every possible combination with every possible inflected form of the adjectives and determiners, thus containing all potential mistakes that can be made but also all correct combinations. Prepositions do not vary, and we also keep the noun stable because it serves as our base word. The alternative phrases are thus created by combining variations of determiner types and adjectives. To produce these phrases PIG goes through the following steps:

- Look up the word in the wordlist and extract the lemma
- Search for all words within the lexicon that have the same lemma and tag
- Generate all potential phrases by combining the alternatives for each word with the alternative for each next word, etc.

Depending on the number of words in the phrase and their possible variations, the number of alternative phrases varies from 4 till above 30. An example:

original input	alternatives:
das gestohlene Auto	des gestohlene Auto dem gestohlenem Auto dem gestohlenen Auto die gestohlenen Auto der gestohlenem Auto der gestohlenen Auto das gestohlenen Auto den gestohlenen Auto dem gestohlene Auto den gestohlene Auto der gestohlene Auto des gestohlene Auto …

3.5. Step 4: Google queries

When creating the queries, PIG constructs a URL for the Google search, using the basic Google search URL, and several modifiers for language:

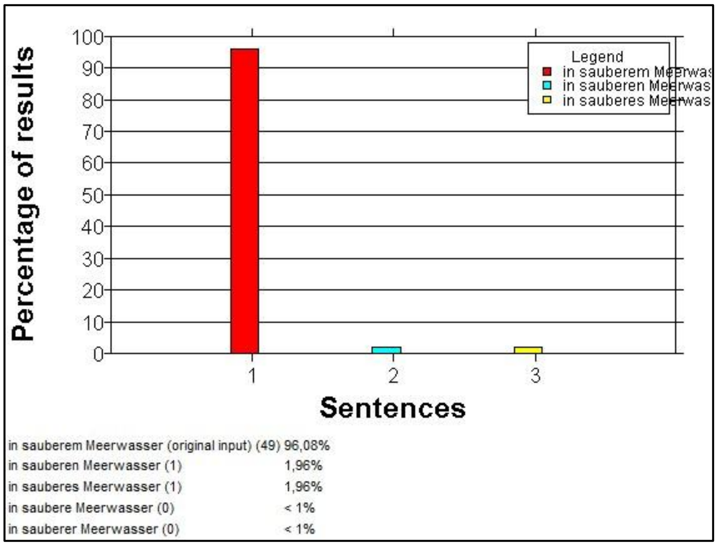
We call the following URL:

`http://www.google.com/search?hl=de&lr=lang_de&q=` + the string to search. The *`hl=de`* and *`lr=lang_de`* specify we wish German results; *`q=`* marks the term to search.

We surround the text to search with quotes, to get the results with the entire string, and not partial matches of the string. Calling the URL will give us the resulting HTML page, from which we extract for the number of results, using a complex regular expression to find the exact characters we need.

3.6. Step 5: Processing of the results

We calculate the total number of results for each phrase set, to be able to calculate percentages. We sort the searched terms by their percentages in descending order. Using a Java plug-in for graph generation, we show the relevant results in a bar chart in an HTML page. Every term with a percentage higher than 1% will be added to the data for the graph, and all the data will be added to the table that is shown underneath it.



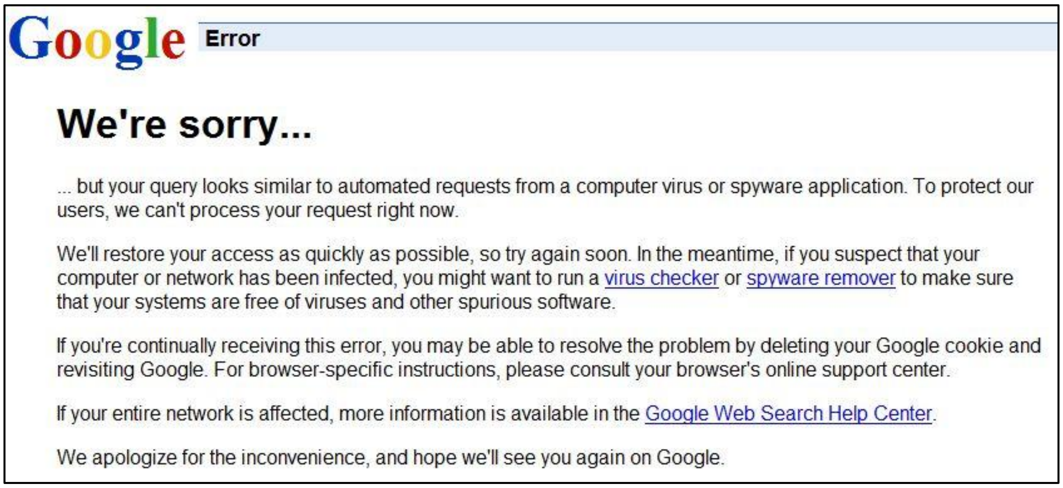
in sauberem Meerwasser (original input) (49)	96,08%
in sauberen Meerwasser (1)	1,96%
in sauberes Meerwasser (1)	1,96%
in saubere Meerwasser (0)	< 1%
in sauberer Meerwasser (0)	< 1%

4. Technical Problems

In this section we deal with the technical problems we faced during the development of the program. The discussion about the challenges regarding to the performance will be part of the evaluation process.

4.1. Google policy

After the first few tests of PIG (approximately 350 URL calls), we encountered the following error message:



We're sorry...

... but your query looks similar to automated requests from a computer virus or spyware application. To protect our users, we can't process your request right now.

We'll restore your access as quickly as possible, so try again soon. In the meantime, if you suspect that your computer or network has been infected, you might want to run a virus checker or spyware remover to make sure that your systems are free of viruses and other spurious software.

If you're continually receiving this error, you may be able to resolve the problem by deleting your Google cookie and revisiting Google. For browser-specific instructions, please consult your browser's online support center.

If your entire network is affected, more information is available in the Google Web Search Help Center.

We apologize for the inconvenience, and hope we'll see you again on Google.

This, obviously, presented us with a problem. Looking in the Google Terms of Service, we found the following statement:

> *Sending automated queries of any sort to Google is against our Terms of Service. This includes, among other things, the following activities:*
>
> - *Using any software that sends queries to Google to determine how a website or webpage ranks on Google for various queries*
> - *'Meta-searching' Google*
> - *Performing 'offline' searches on Google*
>
> *Once you detect and remove the source of automated querying, the ban on your IP address will automatically be lifted within a short time.*

Basically, our automated searches are not allowed, and resulted in our IP address being banned from Google. Since we focused on searching with Google, we just noted this as a problem we cannot fix.

After the completion of the first version of PIG we had an opportunity to consult other research projects that make use of automated queries. According to their experience, Yahoo does not have the regulations Google does concerning the volume and origin of the searches. Therefore, Yahoo might be a better search engine to use for our purpose. The adjusting of the script to search Yahoo would be a next step in the further development of PIG. Given the modular nature of the program, this would not be a complex addition.

4.2. The numbers of Google

Another issue we faced with the Google search engine was rounding. Google is a great way to get an approximation of the number of occurrences in our corpus, the internet (specifically, German language). However, when the number of results generated by a search exceeds a certain amount (5 digits or more), Google rounds the number to the nearest large number. Thus, a search with 654 results will actually return the value 654, but a search with 654321 results will return the value 654000, or 654300.

Another important consideration when using Google as a corpus is the fact that it is constantly changing. The number of hits for a certain phrase can change from day to day, which means that the list of suggestions provided by PIG can also be different from day to day. This should not pose a problem because we are not dealing with individual occurrences but with the relation of numbers of occurrences neither rounding, nor fluctuation of hits should have a significant impact. The only exception would be results where data is really sparse, but one could question the reliability of this data in any case. It might be an interesting test to run the system every seven days for a certain amount of time on a list of phrases to see how much the Google results fluctuate over time, but this is not within the scope of this paper.

5. Evaluation and comparison

5.1. Test data

When selecting our data for testing, our choice fell on exercises from German textbooks, because of two main reasons. First, this data contains a wide variation of adjective phrases that PIG is dealing with. Secondly, when we aim at detecting agreement mistakes, we inevitably define second language learners as our main target group, so a text created for them seems suitable as a test data for the program. The first part of the data is a coherent text taken from[3], the second part are two times ten individual sentences originating from the textbook "Duitse spraakkunstoefeningen" by Vanacker and Timperman[4]. The test data did not undergo any editing process apart from being made suitable for the input. In appendix C we attached a sample of the evaluation charts to have a better understanding of how the comparison with the rule-based grammar checkers was done.

5.2. Error analysis

We developed an error classification in order to distinguish between errors coming from different sources. We suggested a number of bug fixes and prepared the results for comparison with other grammar checking tools. The next chart shows an overview of our error classification.

Error classification		
A. errors in the generator	1. possible combinations not detected in the data	
	2. errors of the tagger or the lexicon:	1. no alternatives found
		2. not all alternatives found
		3. false alternatives found
B. zero hits		
C. errors in performance	1. right combinations marked as incorrect	
	2. wrong combinations marked as correct	

A. Errors in the generator

In this section we deal with errors that occur before feeding the combinations into Google.

A.1 Possible combinations not detected in the data

We manually marked the adjective phrases in our test data that PIG was supposed to extract. We found a hundred percent match when comparing them with the phrases that were detected. This means that all the necessary elements were tagged correctly.

A.2 Errors of the tagger or the lexicon

However, when the alternative phrases were investigated, a few inconsistencies were found mainly due to errors in the lexicon.

A.2.1 No alternatives found

Out of the 58 adjective phrases found in the test data, PIG was able to generate alternatives for 56. In two cases, the lexicon did not find variations. The reason was that the lemma of the words used as attribute were unknown to the tagger, thus each variation was annotated with itself as a lemma. Alternatively, there are cases where the tag for a word and its alternatives seem to differ. Therefore, the program did not find the variations in the lexicon.

The simplest way to deal with this would be to automatically generate the possible word endings, so that it will work with every word, however, exceptions are common in German, and we would effectively be putting a grammatical function into our program.

[3] Dreyer, H./Schmitt, R.: Lehr- und Übungsbuch der deutschen Grammatik. Ismaning/München 2000. p.223

[4] source: http://www.passito.be/index_bestanden/duits/adjektivdeklination04.htm and http://www.passito.be/index_bestanden/duits/adjektivdeklination05.htm, accessed on: 17.01.2009

As this is basically a data sparseness problem, there is not much we can do about it. We can increase the lexicon, to ensure that alternatives for these words are actually in there. We can switch to another tagger, one that might work better, but this is speculative.

We might write a searching function, to find alternatives without using the lemma (looking for words which differ only slightly, but have the same tag or have the same lemma, but not necessarily both, but this might come up with words that are not alternatives. Of course, this last issue might not be a problem, as Google will most likely give no hits for these abnormal combinations. Therefore, this function should be the best solution for this problem.

A.2.2 Not all alternatives found

In one case, some possible alternatives were missing because the word “eine” was tagged incorrectly in the lexicon. This bug has been fixed since. Going over all the closed word classes (thus mainly determinants) in the lexicon to make sure they carry the part-of-speech tag that is most likely to be used in attributive phrases would ultimately filter out this kind of error. Alternatively, we could duplicate them with every possible tag, which will increase the corpus size, but we will be sure that one will be matched.

A.2.3 False alternatives found

In two cases, some incorrect alternatives were created because the comparative of the adjective (carrying the same lemma) was taken along as a variety of ending. At the moment, we use a very basic length difference check to avoid most of these. However, this is inadequate in some cases. An improvement for this can most likely be implemented without adding too much grammatical knowledge. The false alternatives were deleted for the evaluation in order to give fair chances to the two other grammar checkers.

B. Zero hits

When PIG finds no matches on the internet to any of the variations in a set of phrases, it can give no information about the correctness of the input phrase. This is a type of error we can have the least influence on. An additional rule could filter out some of these cases, for example by making PIG check on the results before processing them, and if no hits were found, try to replace part of the phrase by another word that behaves the same (for example cardinals, different determinants e.g.: *zweiter – dritter, ein-kein, sein-mein*). Out of the 58 phrases, 3 resulted with zero hits. We had to take these out of the evaluation chart when comparing with the other grammar checks, because it would give us undeserved credit (while getting an error point for one correct phrase marked as erroneous, we would also get many credits for all the incorrect phrases being marked as zero hit and thus also erroneous). Therefore, we decided to note that according to the test data, PIG has an effectiveness of 94.8% in being able to say something about a phrase. Our test was too small for this to be really significant. The phrases for which this problem occurred were exceptional, and we would have to carry out a test of greater volume to be able to give a good estimate of how many much percentage of the cases would not occur regularly.

C. Errors in performance

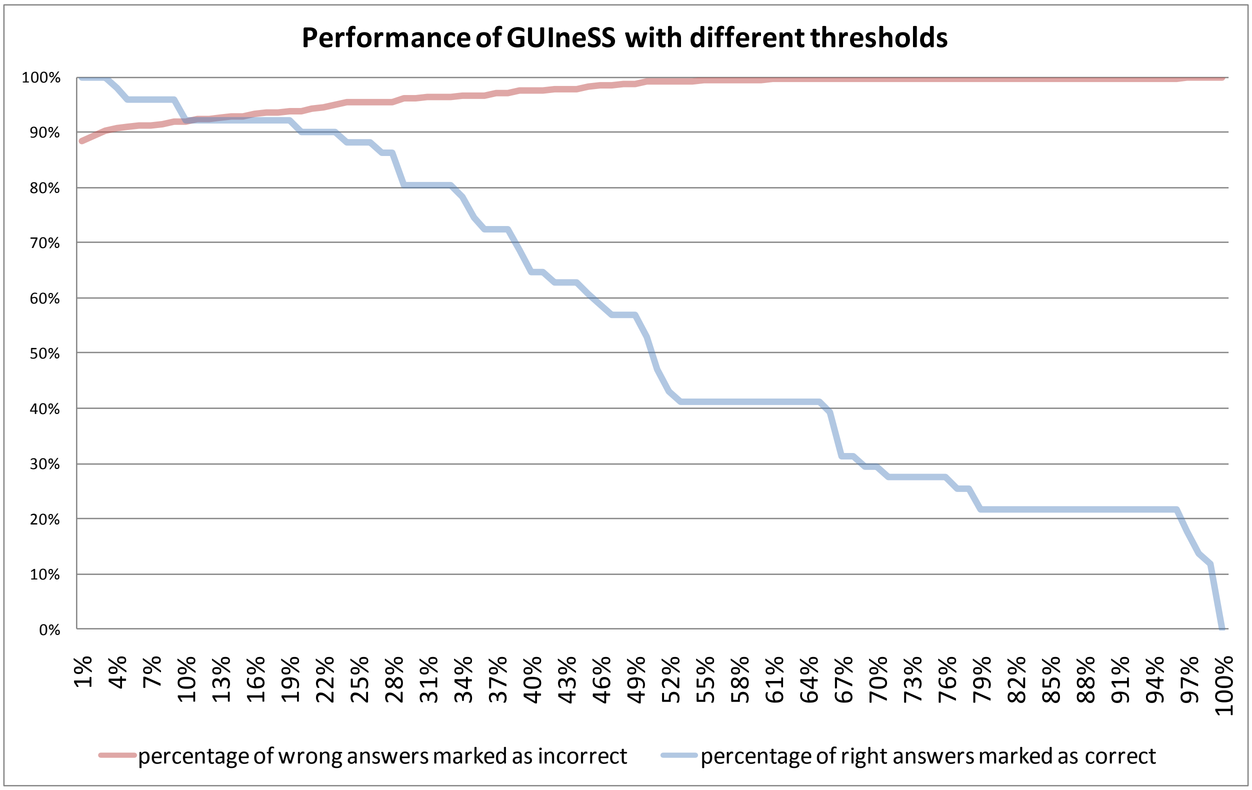
The performance errors will be dealt with in the next chapter.

5.3. Performance evaluation of PIG

In this sequence, we show some charts that calculate the effectiveness of PIG, provided that we assume that only the original input is correct, and everything else is false.[5] According to this assumption, we distinguish between 2 types of errors: right combinations marked as incorrect and wrong combinations marked as correct. The next chart shows how PIG performs with different thresholds of percentage of occurrence. The

[5] For the remaining sections of the evaluation, we will not necessarily follow this assumption.

lower the threshold at which we say, we will suggest this phrase, the smaller the number of correct phrases being filtered out. At the same time, we might end up suggesting phrases that are not correct.



We choose to stay at a threshold of 1%, which means we suggest every phrase that occurs more than 1% compared to other alternatives, as an option for the user. As the next chart shows, we divided the phrases to three different groups, to see whether PIG performs differently with different types of adjective phrases.

Type POS combination	Average percentage of the correct phrase	Average percentage of correct phrase being the highest	Percentage of hits of errors under 1%
A adjective+noun	37.5%	46.2%	29.0%
B phrases starting with a preposition	72.2%	88.9%	81.4%
C phrases with determiner type+adjective+noun	53.2%	75.0%	83.2%
TOTAL	55.9%	72.5%	68.7%

The first column shows the average percentage of occurrence of the original phrase. The second column shows the percentage of phrases being corrected rightly provided that we have a user that always chooses for the suggestion with the highest percentage of occurrence. The third column shows what percentage of the errors was completely filtered out. According to this chart, we can say that PIG performs much better with phrases that contain a preposition. The reason for this is that the presence of the preposition gives a much better clue on estimating the case in which the selected phrase might be in the sentence.

As a final statement, let us look at a fictional case of a user who decides not to bother with PIG`s suggestion list, but always take the combination with the highest number of hits. Such a user would get 72.5% of his sentences right with the help of PIG. This relatively high percentage is partly due to a likelihood issue: a certain phrase that has many occurrences on the web is also more likely to appear to be the correct combination in the test data. On the same line of the argument though, mistakes that do not or very scarcely appear on the web (thus the ones PIG definitely filters out) are also less likely to be made by a user.

5.4. Performance comparison with rule-based grammar checkers

With a range of other grammar checkers on the market, we selected two to compare PIG with. The grammar checkers we selected are the German language spell-and grammar checker of Office Word 2007 and the grammar correcting software Duden Korrektor Plus 5.0[6], developed at the language technology department of Duden.

We generated test data for the two grammar checkers by submitting every alternative PIG generated back into the sentence, gathering all these sentences (1006 sentences before error removal as specified in 0 ; 884 sentences afterwards), and having each grammar checker evaluate these sentences.

Below you will see a table of the number of sentences marked as correct, and the number of sentences marked as incorrect, for each of the grammar checkers.

Duden:		Word2007:		PIG:	
correct	incorrect	correct	incorrect	Correct	incorrect
240	644	386	498	147	737

From this table alone, we can derive several things.

- Word2007 is obviously the least critical of the three, with 45.7 percent marked as correct.
- PIG, using a 1% threshold, is the most critical of the three, with only 15.7 percent marked as correct.
- Duden is in between, with 28.8 percent marked as correct.

Now, given these numbers, we cannot know whether PIG is too critical, or the others are too lenient. Next, we will compare the grammar checkers with each other, in pairs.

Word2007 & Duden:

	Both correct	Both incorrect	Word2007: incorrect Duden: correct	Word2007: correct Duden: incorrect
Result	156	414	84	230

Agreement	Disagreement
0,6447964	0,355204

Here we can clearly see the differences between Word2007 and Duden, as 230 of the 884 sentences are classified as correct by Word2007, and incorrect by Duden. But apparently, there are also cases where Word2007 is more strict than Duden.

[6] For further information please refer to: http://www.duden.de/deutsche_sprache/index.php?nid=113&flip=113 accessed on: 2.2.2009

PIG & Word2007:

	Both correct	Both incorrect	Word2007:incorrect PIG: correct	Word2007:correct PIG: incorrect
Result	115	466	32	271

Agreement	Disagreement
0,6572398	0,34276

Again we can see the difference in strictness. Noticeably, like with Duden, there are a few cases that PIG classifies as correct, while Word2007 claims they are incorrect.

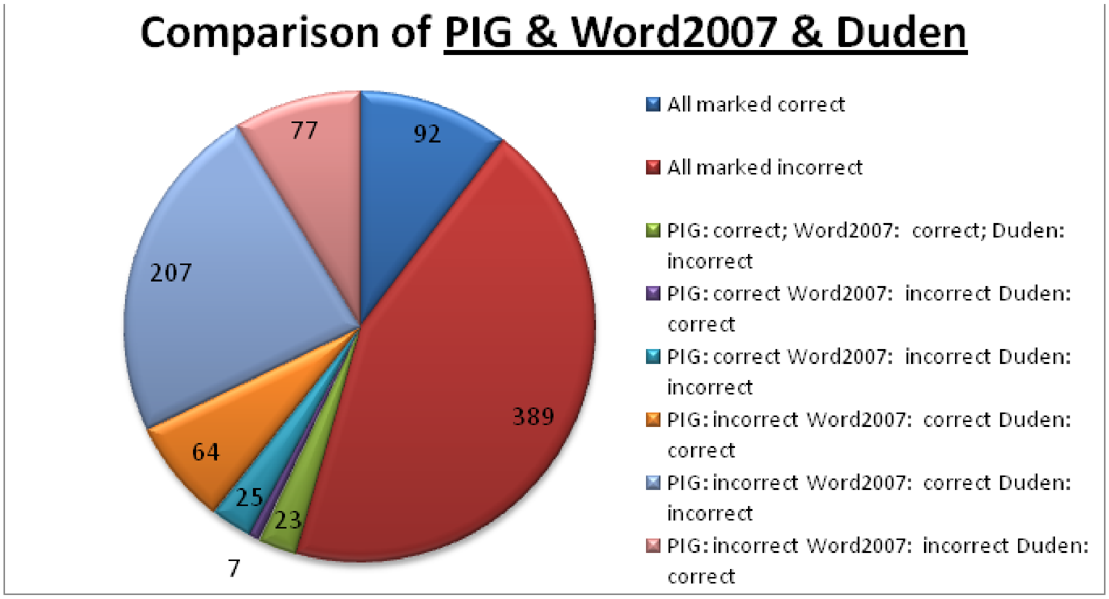
PIG & Duden:

	Both correct	Both incorrect	PIG: incorrect Duden: correct	PIG:correct Duden: incorrect
Result	99	596	141	48

Agreement	Disagreement
0,7861991	0,213801

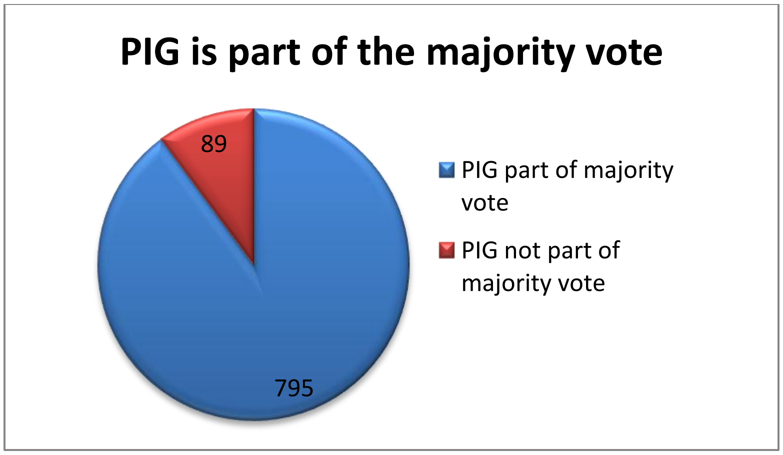
Here we see the agreement between PIG and Duden is a lot higher than the agreement of either with Word2007. This makes sense, they are both a lot more strict than the Word2007 grammar checker. Here, too, are cases where the less strict grammar checker has classified sentences as incorrect, while the more strict one has not. In the next chart we will compare all three.

PIG & Word2007 & Duden:



Agreement	Disagreement
0,5441176	0,455882

We see from the agreement that there are a lot of cases where there is some disagreement between the three grammar checkers. For analysis, we will assume that when two out of three agree on a classification, that classification is most likely to be true.



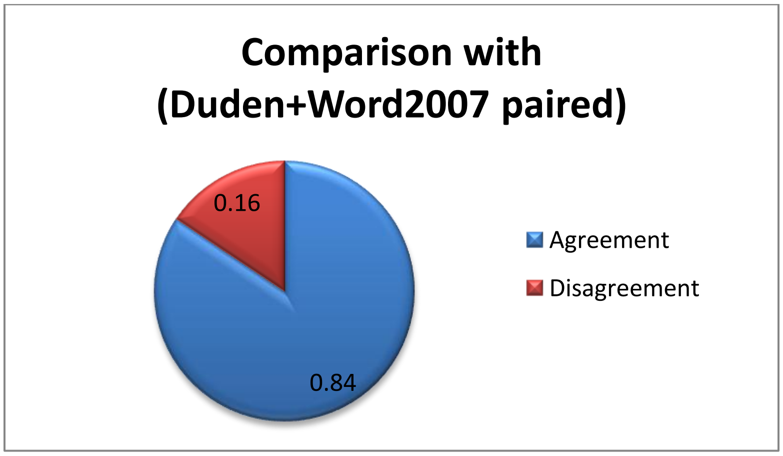
PIG part of majority vote	PIG not part of majority vote
795	89

Now, this looks promising. This would imply that in 90% of the cases, PIG was right, by our previous assumption. Of course, this could also mean PIG is consistently siding with the wrong result, so we will have to look at it more closely.

Accuracy of PIG where Word2007 and Duden agree:

This is a good way to exclude strange behavior from Word and Duden in our comparison, as we can see in the table above that a large part of the disagreement between the three grammar checkers is because of incorrect classifications by the Word2007 grammar checker.

Total of agreed values between Duden and Word2007: 570



We can assume these are an accurate representation of the strength of using both grammar checkers combined, as these are the cases they agree. Given these values, PIG gives the same result for 481 of them. This means there is a 0,843859649 agreement between PIG and the others, if Word2007 and Duden are seen as one grammar checker.

Specifically, there are only 25 cases where PIG marked something as correct while both Word2007 and Duden agreed it is incorrect, and there are 64 cases where PIG marks something as incorrect while Word2007 and Duden agree it is correct.

The other 314 are cases where Word2007 and Duden disagree:

Where Duden and PIG agreed (thus according to our assumption above, were right) and Word2007 disagreed:

- 7 cases where Word2007 said incorrect, while the others said correct
- 207 cases where Word2007 said correct, while the others said incorrect

The above shows how unreliable (lenient) the Word2007 grammar checker is compared to the other two, since it allows far more combinations. Specifically, we can see that 207 of the 314 cases where Duden and Word2007 disagree are cases where the Word2007 grammar checker is not strict enough. We already saw, in our basic comparison, that the Word2007 grammar checker marks a significantly higher amount of the cases as correct, while Duden and PIG clearly agree that these are incorrect.

Where Word2007 and PIG agreed and Duden disagreed:

- 77 cases where Duden said correct, while others said incorrect
- 23 cases where Duden said incorrect, while others said correct

Knowing what we now know of the Word2007 grammar checker, the 23 cases are rather dubious. These 23 cases may very likely be faults of PIG (as Word2007 agrees with half of the input anyway), which can mostly be explained by an incorrect combination getting just enough Google results to get above the 1% used for this classification.

The 77 cases, though, are more interesting. Duden appears to allow these, while they are quite likely incorrect. Word2007, weak as it is, even manages to mark these as incorrect, and they apparently have an insignificant or even zero amount of results on Google.

To summarize, both when we compare our results with the paired results of Duden and Word2007 (which will most likely be reliable for correctness), and when we look at PIG in the majority vote (which is less reliable for actual correctness, but more reliable for agreement with either or both) we can see that PIG is quite close to the results of the other two grammar checkers. For pure comparison the majority vote will probably be the most significant.

5.5. Performance evaluation with phrases out of context

In this last section of our evaluation we look at the correctness of the generated phrases out of context. This means, we distinguish between combinations that are possible in other sentences (with a different case for example) and the combinations that are not possible at all. To achieve this, we asked a native speaker to mark our entire evaluation data, judging the alternatives for grammaticality.

After that we compared agreement between the grammar checkers and the manual judgments to find the following data:

PIG	Office Word 2007	Duden Korrektor Plus 5.0
96.0%	66.7%	81.2%

These seemingly impressive results are partly due to the high number of erroneous phrases as opposed to grammatical ones. And detecting errors is what PIG is performs well in. At the same time, if we look at the suggestions PIG gives, (2.9 suggestions on average) we only find a correctness of 76.2%. This means, almost 1 out of 4 of PIG`s suggested phrases are not at all possible, in no other context either. The evaluation of the suggestions of Office Word has not taken place within this project, because the comparability with PIG is questionable, not knowing if Office Word takes context into account. Duden Korrektor does not give any suggestions when incorrect phrases are encountered, only a warning message for further consideration. To conclude, PIG does sort out 95.5% of the incorrect combinations, but is apparently less reliable when it comes to suggesting a correct phrase.

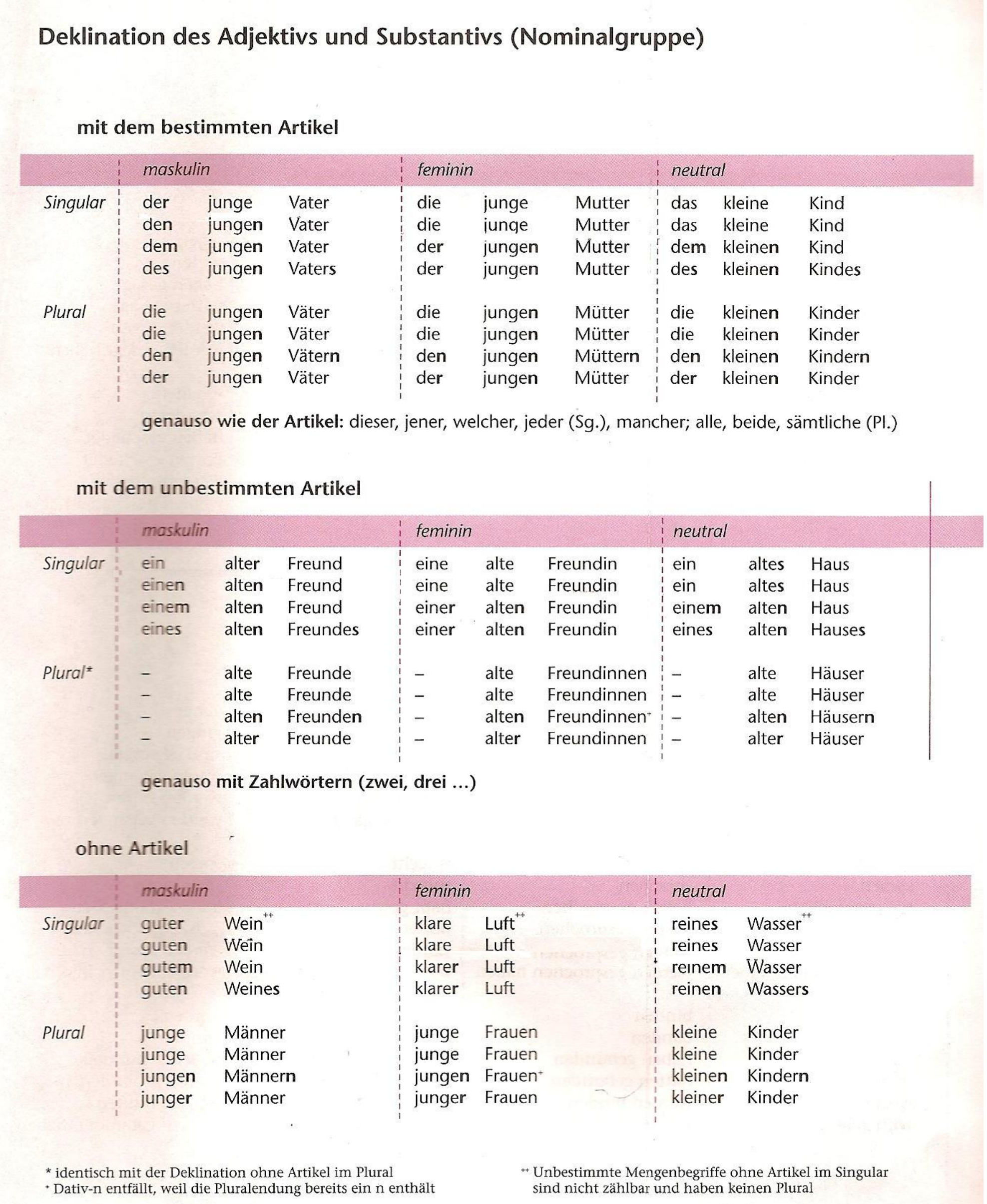
6. Conclusions

To summarize, let us revisit our initial hypothesis, according to which “a grammatically correct phrase will in general appear significantly more often than the same phrase containing an error”. When evaluating PIG we were faced with the likelihood issue, according to which PIG better at correcting mistakes that are unlikely to appear, than errors that are very likely to be made, because those might appear on the web as well. However, the hypothesis has proven to be right, considering that 95.5% of the erroneous phrases were filtered out by PIG.

We can conclude that even though there are several considerations to be made, within the field it is aimed at, PIG seems to approach the quality of the two rule-based grammar checkers. From this, we can assume that the GUIneSS concept might be viable, though it remains to be seen whether it will work as well with other combinations.

7. Appendix

A) Adjektivdeklination



Deklination des Adjektivs und Substantivs (Nominalgruppe)

mit dem bestimmten Artikel

	maskulin			*feminin*			*neutral*		
Singular	der	junge	Vater	die	junge	Mutter	das	kleine	Kind
	den	jungen	Vater	die	junge	Mutter	das	kleine	Kind
	dem	jungen	Vater	der	jungen	Mutter	dem	kleinen	Kind
	des	jungen	Vaters	der	jungen	Mutter	des	kleinen	Kindes
Plural	die	jungen	Väter	die	jungen	Mütter	die	kleinen	Kinder
	die	jungen	Väter	die	jungen	Mütter	die	kleinen	Kinder
	den	jungen	Vätern	den	jungen	Müttern	den	kleinen	Kindern
	der	jungen	Väter	der	jungen	Mütter	der	kleinen	Kinder

genauso wie der Artikel: dieser, jener, welcher, jeder (Sg.), mancher; alle, beide, sämtliche (Pl.)

mit dem unbestimmten Artikel

	maskulin			*feminin*			*neutral*		
Singular	ein	alter	Freund	eine	alte	Freundin	ein	altes	Haus
	einen	alten	Freund	eine	alte	Freundin	ein	altes	Haus
	einem	alten	Freund	einer	alten	Freundin	einem	alten	Haus
	eines	alten	Freundes	einer	alten	Freundin	eines	alten	Hauses
*Plural**	–	alte	Freunde	–	alte	Freundinnen	–	alte	Häuser
	–	alte	Freunde	–	alte	Freundinnen	–	alte	Häuser
	–	alten	Freunden	–	alten	Freundinnen+	–	alten	Häusern
	–	alter	Freunde	–	alter	Freundinnen	–	alter	Häuser

genauso mit Zahlwörtern (zwei, drei ...)

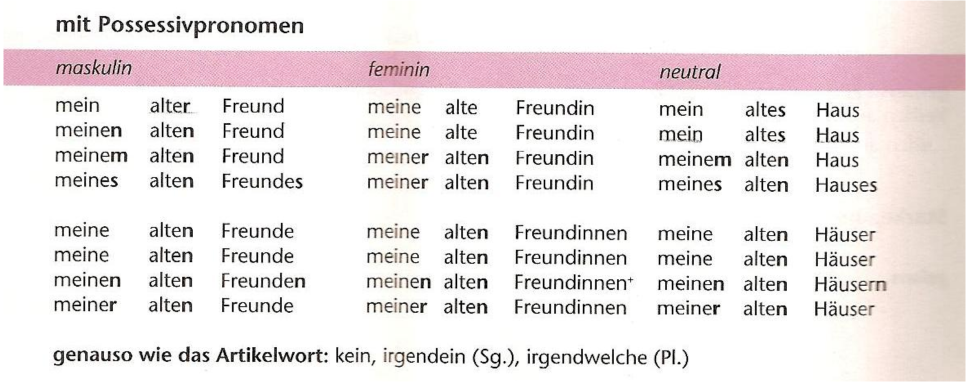
ohne Artikel

	maskulin		*feminin*		*neutral*	
Singular	guter	Wein++	klare	Luft++	reines	Wasser++
	guten	Wein	klare	Luft	reines	Wasser
	gutem	Wein	klarer	Luft	reinem	Wasser
	guten	Weines	klarer	Luft	reinen	Wassers
Plural	junge	Männer	junge	Frauen	kleine	Kinder
	junge	Männer	junge	Frauen	kleine	Kinder
	jungen	Männern	jungen	Frauen+	kleinen	Kindern
	junger	Männer	junger	Frauen	kleiner	Kinder

* identisch mit der Deklination ohne Artikel im Plural

+ Dativ-n entfällt, weil die Pluralendung bereits ein n enthält

++ Unbestimmte Mengenbegriffe ohne Artikel im Singular sind nicht zählbar und haben keinen Plural



mit Possessivpronomen

maskulin			*feminin*			*neutral*		
mein	alter	Freund	meine	alte	Freundin	mein	altes	Haus
meinen	alten	Freund	meine	alte	Freundin	mein	altes	Haus
meinem	alten	Freund	meiner	alten	Freundin	meinem	alten	Haus
meines	alten	Freundes	meiner	alten	Freundin	meines	alten	Hauses
meine	alten	Freunde	meine	alten	Freundinnen	meine	alten	Häuser
meine	alten	Freunde	meine	alten	Freundinnen	meine	alten	Häuser
meinen	alten	Freunden	meinen	alten	Freundinnen+	meinen	alten	Häusern
meiner	alten	Freunde	meiner	alten	Freundinnen	meiner	alten	Häuser

genauso wie das Artikelwort: kein, irgendein (Sg.), irgendwelche (Pl.)

B) Pseudo-code

```
Dictionary Dict = new Dictionary()
GoogleSearch Google = new GoogleSearch()
HTMLResult Html = new HTMLResult()
PredefinedThreshold = 1%

// Step 1: TOKENIZATION AND TAGGING
Text = Read text from input or file

// Prepare text for tagging
Text = replace(“\s”, “\n”)
Text = replace(“,”,“”)
Text = replace(“[\.!\?]”,”\n.”)

// Tag text
external_call(wintreetagger.exe)

// Step 2: DETECTING N-GRAMS

Wordgroups = get_all_matching(
“(APPR)?(PPOSAT|ART|PDAT|PWAT|PIAT|PIS|CARD|PDS)?(ADJA)+(NN)”,Text)

// For each word group:
for(Wordgroups as Wordgroup)

      // Step 3: GENERATING ALTERNATIVE PHRASES
      Alternatives = Dict.generate_alts(Wordgroup)

      // Step 4: GOOGLE QUERIES

      // Google search Wordgroup
      Results[Wordgroup] = Google.search(Wordgroup)

      // Google search each alternative
      foreach(Alternatives as Alt)

            Results[Alt] = Google.search(Alt)

      // Step 5: PROCESSING OF THE RESULTS

      // Calculate total count by summing all results
      Total = sum(Results)

      // Calculate percentage of Total and sort the list
      Percentages = sort(apply(Results, divide_by(Total)))

      // Write the results to the html table; write all with more than
      a predefined percentage to the graph
      Html.graph(Percentages, PredefinedThreshold)
      Html.table(Percentages)
```

C) EVALUATION CHARTS

33	GUIneSS		Microsoft Word 2007	Duden Korrektor
	amount	percentage		
durch die chemische Industrie (original input)	-1880	99.84%	-	C
durch die chemischen Industrie	-2	0%	X chemische	X
durch der chemischen Industrie	-1	0%	X die chemische	X
durch das chemische Industrie	0	0%	X die	X
durch dem chemische Industrie	0	0%	X die	C
durch den chemische Industrie	0	0%	X die	X
durch der chemische Industrie	0	0%	X die	X
durch des chemische Industrie	0	0%	X die	X
durch das chemischem Industrie	0	0%	X die chemische	X
durch dem chemischem Industrie	0	0%	X die chemische	X
durch den chemischem Industrie	0	0%	X die chemische	X
durch der chemischem Industrie	0	0%	-	X
durch des chemischem Industrie	0	0%	X die chemische	X
durch die chemischem Industrie	0	0%	X chemische	X
durch das chemischen Industrie	0	0%	X die chemische	X
durch dem chemischen Industrie	0	0%	X die chemische	X
durch den chemischen Industrie	0	0%	X die chemische	X
durch des chemischen Industrie	0	0%	X die chemische	X
durch das chemischer Industrie	0	0%	X die chemische	C
durch dem chemischer Industrie	0	0%	X die chemische	X
durch den chemischer Industrie	0	0%	X die chemische	X
durch der chemischer Industrie	0	0%	X die chemische	X
durch des chemischer Industrie	0	0%	X die chemische	X
durch die chemischer Industrie	0	0%	X chemische	X
durch das chemisches Industrie	0	0%	X die chemische	X
durch dem chemisches Industrie	0	0%	X die chemische	X
durch den chemisches Industrie	0	0%	X die chemische	X
durch der chemisches Industrie	0	0%	-	X
durch des chemisches Industrie	0	0%	X die chemische	X
durch die chemisches Industrie	0	0%	X chemische	X
34	GUIneSS		Microsoft Word 2007	Duden Korrektor
	amount	percentage		
verschmutzten Fluss	-479	41.44%	-	C
verschmutztem Fluss	-252	21.80%	-	C
verschmutzter Fluss (original input)	-224	19.38%	-	C
verschmutzte Fluss	-192	16.61%	-	X
verschmutztes Fluss	-9	0%	X verschmutzter	X