NLP, NLU, Language related AI SYSTEM
What our Language Processing Technology can do

Our system can find **implicit** (tacit, hidden, unseen) **information** in the Natural Language sentence. It is a patented invention, with developed working prototypes for English and German. In order to find the implicit information, it finds first the **explicit information**, the information expressed with actual words. The explicit information is on all Language levels: **morphological** (composition of the word), **grammatical** (gender, number, case, etc.), **syntactical** as Part of Speech (Noun, Adjective, Adverb, Preposition, Verb, incl. Verbal Tenses, etc.) and as Part of the Sentence (subject, object, complement, etc.), **semantical** (the meaning of the word in particular context).
We have developed efficient syntactical and semantical disambiguation procedures, capable to analyze each ambiguous word entry in context.
Our system is not programmed to find pragmatical information, because if something for somebody has a pragmatical value, the same thing, for somebody else, has no value. However, since the system understands the explicit and the implicit meaning of the sentence, it can be programmed to find information with pragmatical value for certain groups of people.
As soon as we leave the boundaries of the Sentence and start to talk about text(s), things look different. A text has a general meaning, this meaning can be attributed to a particular **subject field, theme of discourse, topic**, which our system does. Our system can calculate also the percentage of each subject field, present in the text, the percentage of each theme or topic, even the percentage of each word meaning, if needed. We have shown, in an academic publication, that there are around 5,000 word meanings common to all Languages, present in 40-50 thousand most used words.

Our system, some call it engine, is written in C/C++. The system works with coded information. The codes represent different morphological, grammatical and semantical features present in the text, in the sentence, in the phrase and in the word(form). The codes are declared with programming code and entered, respectively, in the database. In our case, the database is an alphabetically arranged dictionary, consisting of words (wordforms) and phrases, the phrases can be idioms, standard expressions, adverbial phrases, verbal phrases (verb used with a particle or a preposition).
All phrases have a distinctive and inseparable meaning. There is no limit of word length or phrase length. For example,
*deckbett*N[coded information]*
The sign after the asterisk means Noun.
*degustieren*V[coded information]*
The sign after the asterisk means Verb, etc. for all Parts of Speech.

If the word is a typical misspelling it can be also entered in the dictionary and changed with correct spelling each time the misspelling is used, as we have done in the spell-checkers.
For example
*anwarmen*V[ ]anwärmen*
*apelkahn*N[M]appelkahn*
*appelkahn*N[M]appelkahn*
where [M] means masculine gender.
The word on the left can be a word from the source language and the word on the right can be a corresponding in meaning and context translation(s) in the target language, in case of Machine Translation from one language into another. In one of our software programs, English-German text-to-voice Machine Translation you write in English and hear the translation spoken in German using the correct case Article, etc.
When the word is ambiguous, i.e. belongs to more than one Part of Speech, it is also coded respectively and triggers a procedure, in the system, to analyse the context and to determine exactly to which Part of Speech the word belongs in this particular context. For example
*braeune*Z[0]/N[0]/A[\]*
is ambiguous, marked with Z for Verb, N for Noun and A for Adjective.

All this is explained in great detail in the book "Language Engineering", written for those who know and do not know programming. For a programmer, proficient in C/C++, there will be nothing unknown when reading our programming code. All codes are declared.
What exactly our system can do, is a trade secret. The competition finds it out by downloading and testing our timed commercial versions offered for evaluation.
Some of these secrets and possible future applications are listed below.

GENERAL WORD and SENTENCE ANALYSIS, APPLICABLE TO ALL TEXT PROCESSING TASKS

A) Word analysis.
From now on the term "word" will be used to mean word and wordform (the wordforms are the various endings of the same word). The term "wordform" will be used only when one of the grammatical forms of the same word is meant.
1. Our NLP system can cut the ending of any word, the ending to be cut is specified, for example *-e, -ing, -ion,* etc. and replace the cut ending with another ending, a procedure needed in word recognition, when the entry wordform is not registered in the dictionary. For example, if we have registered, in the dictionary, the word "*abbreviate*", but not registered, in the dictionary "*abbreviated*", "*abbreviating", "abbreviation*", our system will cut the ending *-e* from "*abbreviate*" and add *-ed or -ing* or *-ion*. This way, our system will recognize the words *"abbreviated", "abbreviating", "abbreviation"*, when met in the text we analyse,

despite the fact, that they are not registered in the dictionary used by the software program. When there is no ending to be cut from the word, registered in the dictionary, then the ending is regarded as zero and the system can add an ending to the registered word, for example "*access*". When we register in the program the endings *-ed. -ing*, the program will recognize, when analysing the text *"accessed", "accessing*", even though they are not registered in the dictionary. Note, that if we add also *-ion*, the program will recognize the word "*accession*", when met in the text, however, here we face ambiguity, because "*accession*" has totally different meaning and does not stem from "access", used as Noun or Verb. Natural Language is full of such ambiguities and we have done our best to minimize or eliminate them. One way to eliminate such ambiguity is to register the word "*accession*", in the dictionary and attach to "*accession*" other semantical codes, different from the codes attached to "*access*".
As far as English is concerned, we had just managed to deal with around 80,000 relevant words, in the past, with all memory restrictions we had to face. Nowadays, we have, practically, no memory restrictions and we can register unlimited number of word and phrase entries in the dictionary.
Note also, that in the morphological analysis of the word, many word endings are ambiguous: they can be the same for an Adjective, for a Noun and for a Verb, etc. We have defined what Parts of Speech what word endings can have, those wordforms, that are ambiguous, we have registered in the dictionary.
What we have said above, about the words and their endings relates to the analysis of the input (source) language. In case of Machine Translation, we have also output (target) language. Our system is capable of applying similar morphological procedure for the target language as well. Namely, we register, as a rule, the basic form of the target word, i.e. the word that has to be written as the equivalent of the input word. However, this basic form of the target word can have many wordforms. Only one of the wordforms is used in a specified context, depending on number, gender, case, verbal tense, etc. Our system is capable to add the appropriate word ending, depending on the agreement of this word with the other words in the sentence in number, gender, case, verbal tense, etc. For example, in German, the Adjective takes the respective endings depending on its agreement with the Noun. There are different Articles, depending on the Case of the German Noun, etc. Our system chooses the correct Article, the correct ending of the Adjective, etc. endings, as needed.

Our system can compare and match word or wordform from the text with specified word or wordform within the system and when the match is found do what it is instructed to do.
Our system can compare and match word ending or wordform ending from the text with word ending or wordform ending specified within the system and when the match is found do what it is instructed to do. For example:

```
if (!stricmp(wrd->inword, "bahn") || isflex(wrd->inword, wrd->wl, "bahn")){
} // etc. instruction what to do when the match is found
```

It does not matter here, whether "*Bahn*" is written with small letter or with capital letter.

An important part of the morphological analysis is the prefix of the word. German has more prefixes than English because German links two or more words to make one word. The first word in linked words is regarded as prefix, because it is used to create many other words. Our system has a procedure to recognize a prefixed word, written in the text we analyse, even when this prefixed word is not registered in the Dictionary. This is also a lengthy procedure to explain how it is done. In short, we have a file with list of prefixes. When our system cannot find the prefixed word in the dictionary, it adds prefixes to compare, until a match is found. Here also are many exceptions that would lead to ambiguity, if not excluded. The best exclusion is to register all prefixed words in the dictionary and leave this rule just in case.
For the purposes of Machine Translation, from one language into another, our system can translate the prefix of one language into prefix of another language, when the prefixed word is not registered in the dictionary, with its translation. For example "*black- Schwarz-*", "*multi vielfach-*", etc., when the English word "*berry*" is registered in the dictionary, but "*blackberry*" is not registered, our system finds "*berry*", with its German translation and adds "*Schwarz*" to the German translation of "*berry*".

Our system can compare and match the initial part of a target language word or wordform from the text with specified initial part of target language word or wordform within the system and when the match is found do what it is instructed to do. For example:

```
if ((wrd->cw[l + 0] == 'p' && wrd->cw[l + 1] == 'u' && wrd->cw[l + 2] == 'n' && wrd->cw[l + 3] == 'i'
&& wrd->cw[l + 4] == 's' && wrd->cw[l + 5] == 'h') || (wrd->cw[l + 0] == 'p' && wrd->cw[l + 1] ==
'e' && wrd->cw[l + 2] == 'n' && wrd->cw[l + 3] == 'a' && wrd->cw[l + 4] == 'l')) {
// the initial parts are "punish" and "penal", the numbers show the letters
// in sequence, for English etc. instruction what to do when the match is found
}

if (wrd->cw[l + 0] == 's' && wrd->cw[l + 1] == 'p' && wrd->cw[l + 2] == 'a' && wrd->cw[l +
3] == 'r' && wrd->cw[l + 4] != 'r') {
// the initial part is "spar" (saving), words with double "r" are excluded
// one can include or exclude any position the same way as it is done above
// on positions that are not excluded or included, any letter can take
// this position
// etc. instruction what to do when the match is found, for German
}
```

Our engine can analyse more than ten consecutive letters, but for matching the initial part of long words, it was not needed, as our practice has shown.

Target language can be also the same language as the source language, as is the case with our spell-checkers, text to voice and video.
Our system recognizes the German separable verbs, in context, and provides accurate recognition of meaning and translation into another language.

B) Syntactical analysis and operations with the words within the sentence.
The syntactical information, within the sentence is present, to a certain extent, in the Parts of Speech. Every language has certain rules of arranging the Parts of Speech within the sentence.
In the book "English Algorithmic Grammar", are published Parts of Speech sequences for the English language. In the book "Language Engineering", are published such sequences for German and French.
By syntactical information we understand the role of the word within the sentence and its relationship(s) with the other words, in the same sentence. It is highly disputable what comes first: whether this arrangement makes the meaning, or the meaning necessitates certain arrangement. Whatever the case, we must deal with it when analysing Natural Language texts.
The Case and the Number are part of these word relationships and result from them. We have defined the syntactical information with source code in the respective files and marked the Parts of Speech in the dictionary, for each word (phrase) entry. The syntactical information is indispensable in sentence analysis. We cannot analyse the grammar and the meaning of the sentence, without it.

1. Our system can analyse more than seven consecutive words. For example
word1 word2 word3 word4 word5 word6 word6+1,
word6+1 is not marked in the *.h file, the last marked word is word6, but we can extend the "if" rule, in the *.cpp file, by specifying "last plus one word" for consideration as well.
In rare cases we have analysed up to 12 consecutive words in a compound sentence.
The consecutive words can be separated by any number of optional words and phrases, by optional we mean that these words and phrases can actually exist in the sentence we analyse or not.
We specify what Parts of Speech are optional, in a rule, in one of our *.h files. For example:
word1 <Adjective>word2
when word1 is an Article and word2 (the last, in this case) is a Noun, the Adjective is optional.
That means, there can be an Adjective, preceding the Noun, or the Noun can be used without preceding Adjective(s). By using such rule, to analyse the input text, such sentences as
"The voice"
"The loud voice"
"The loud imperative voice",

etc. Adjectives between, will be equally recognized when met within the sentence. Note that, in our system, word1 can be any word within the simple sentence or within the compound (complex) sentence. Word2 can be also any word within the simple or compound sentence.
In case of a compound (complex) sentence, only within the beginning and the end of the compound (complex) sentence. Note also, that "voice", in the above example, is ambiguous. It can be a Noun, a Verb or an Adjective. In another underlying rule, preceding all other rules, we have already instructed our system, that in this particular context, after an Article, Indicative Pronoun, Preposition, etc. the ambiguous word, in position word2 can be only a Noun.
For a reference how that is done algorithmically, please, see the algorithms in the book "English Algorithmic Grammar".
In short, word1 can be any word within the sentence and the last word can be also any word within the sentence, with any number of optional, intervening words between them or between word2 and word3, between word3 and word4, between word4 and word5, between word5 and word6.
However, we are very careful when specifying too many intervening words, we better write several separate rules than burden one rule with so many possibilities.
Our engine can analyse more than seven consecutive words, but for the syntactic and semantic analysis of the sentence, it was not needed, as our practice has shown.

2. Our system can re-position the words within the sentence, whether for the purposes of single language (English into plain English, Deutsch ins einfacher Deutsch) to correct style of writing, or wrong word order, or whatever else is needed, or for the purposes of bilingual Machine translation, to change the word order of the source language into the word order rules of the target language. For example, if we translate the English sentence "*He has seen the woman*" into German, our system will re-position the words, in this sentence, in the German translation and the output will be "*Er hat die Frau gesehen*". This we do with a rule, that takes "the woman" and puts it after the Auxiliary Verb and before the Past Participle, as a typical word structure, in German. The same is done, when our system translates from German into English. Our system can move and re-position any word or sequence of words.

3. Our system can insert a word or a phrase, in the sentence, also for the purposes of single language or bilingual Machine Translation, either to correct style of writing or to improve the translation. Our system can insert also a Punctuation Mark. For example, if we have to translate two consecutive Verbs from German into English, such as "... *akzeptieren heisst* ...", in the English translation we have to insert "to" before the first Verb, in order to obtain "... *to accept means* ...". Of course, in such case, the translation can be "... *accepting means* ...", all depends on context.

4. Our system can hide (not display) certain word, also for the purposes of single language or bilingual Machine Translation, either to correct style of writing or to improve the translation. For example, if we translate the German sentence "*Er erinnert sich, dass ...*" into English, the Reflexive Pronoun "sich" will be out of place, therefore the translation of "sich" must not be displayed. Our system does that with the appropriate instruction and the translation will be "*He remembered, that ...*". Likewise, our system can hide any word or phrase, within the sentence, as needed.

5. Parsing is part of the syntactical analysis. Please, see the description of Syntparse below.

C) Semantical Analysis

The semantical information is presented within square brackets, in the dictionary, for each word entry and defined with source code in the respective files. We use, at present, around 195 different semantic codes (ASCII characters of various alphabets) and many code combinations (combination of two codes to mark a meaning, different from the meaning expressed by each of these two codes, when used individually). The Gender is one, coded with one of these codes. For example [M] is Masculine Gender [F] is Feminine Gender, etc. With the codes we mark mainly concepts (bird, fish, animal, tree, vegetation, human being, etc.). We do not use codes for the synonyms.

There are around 4,000 synonymical groups, published in the book "Dictionary of Word Meanings". At present we work with 2,209 German word meanings and 1586 English word meanings, 195 of them are concepts. Different combinations of the meanings provide accurate picture of the content of the written text. We prefer to list the synonyms, as a group, in the source code of the program, in the respective case, dealing with particular meaning. For example:

```
if (!stricmp(wrd->inword, "ordnen") || !stricmp(wrd->inword, "arrangieren") || !stricmp(wrd->inword, "drapieren") || !stricmp(wrd->inword, "strukturieren") || !stricmp(wrd->inword, "tabellarisieren") || !stricmp(wrd->inword, "klassifizieren") || !stricmp(wrd->inword, "relegieren") || !stricmp(wrd->inword, "systematisieren") ||
!stricmp(wrd->inword, "einteilen") || !stricmp(wrd->inword, "terminieren") || !stricmp(wrd->inword, "einstufen") || !stricmp(wrd->inword, "sortieren") || !stricmp(wrd->inword, "anstehen") || !stricmp(wrd->inword, "rubrizieren") || isflex(wrd->inword, wrd->wl, "ordnen") ||
isflex(wrd->inword, wrd->wl, "sortieren") || !stricmp(wrd->inword, "verteilen")) {

 (void)system("ordnen.mp3"); // meaning: arrange
// instruction what to do
// ordnen.mp3 can be arrange.mp3, spoken in English,
// or ordnen.mp3 spoken in any other language, when we translate,
```

```
// text-to-voice, from German into another language
}
```

Because many words have several meanings, while in context they have only one meaning, we have to define this meaning with source code and say, in what context, which meaning is relevant. For example:

```
if (wrd->mood == 1 && (wrd->E.W.feeling == 1 && wrd->E.W.negative == 1 && wrd->tense == T_PRESENT && wrd->person != 3)) {
 (void)system("hassen.mp3"); // meaning: hate
}
```

where "mood" is a pre-defined context, in another rule, the rest are defined semantical codes, Verbal Tense and Person (here we say "not third person“).

Arrange, hate, etc. meanings are language independent. They can be met in any language and can be used for text attribution and text classification of texts written in any language, provided we do the same, for each language, as we have done for English and German.
For ambiguous words we have rules to cancel semantical information assigned in the Dictionary and assign new semantical information, depending on context.

Our Natural Language Text Processing System effectively disambiguates the homophones in context for the needs of voice-to-text-text-to-voice Machine Translation.

I) PARSING (English, German, French, Italian, trademark Syntparse):
1. Recognizes, correctly, all Parts of Speech (Noun, Verb, Adjective, etc.), used in context.
2. Recognizes, correctly, all Parts of the Sentence (Subject, Verbal Tense, Object Direct, Object Indirect, Complement Subject, Complement Object, etc.). We use the traditional sentence analysis, not the modern, offered by Chomsky. For full description, please read our books. Also, you can test our evaluation copies, downloadable from our website.
3. Our parsers can display the concept and the meaning of each word.

II) SPELLING (English, German, French, Italian):
1. Recognizes the orthographically misspelled words and suggests corrections (older versions).
2. Recognizes the orthographically misspelled words and automatically replaces them with correctly spelled words (newer versions for English, German, French).
    If a name is written with a small letter, the program automatically writes it with a capital letter, respectively, the program writes with a capital letter all German Nouns, including those that can be either Noun or Verb, in context. Due to our parsing, the program disambiguates the Parts of Speech in context.

3. Recognizes the grammatically incorrect words (agreement in number, gender, case, the use of certain Prepositions only with certain words, the use of the German sein and haben only with certain Past Participles, etc.) and suggests corrections (older versions).
4. Recognizes the grammatically incorrect words (agreement in number, gender, case, the use of certain Prepositions only with certain words, the use of the German sein and haben only with certain Past Participles, etc) and automatically replaces them with correctly spelled words (newer versions for English, German, French).
5. The German version corrects automatically all irregular Nouns (singular - plural forms) and irregular Verbs.

Voice recognition is not 100% correct. The errors are around 25%. If one wants to translate voice-to-voice or voice-to-text from one language into another, the errors will increase up to 40% or even more, because the translation will be based on erroneous, misunderstood or missing parts of the sentence. Our system, with the help of our semantical and grammatical analysis of the sentence, can re-construct the missing parts and correct what is erroneous or unrealistic in the meaning. The translation errors will be reduced to a great extent. The product will be commercially viable.

III) MACHINE TRANSLATION (English-German, German-English):
1. Chooses the correct word equivalent, depending on context and meaning. There are several procedures to choose from when selecting the best equivalent, specially when dealing with ambiguous words. All these procedures have one thing in common: they fix the context in which the ambiguous word has only one meaning. Often the translation equivalent depends on the meaning of the sentence, on the concept the word belongs to.
For example "to hang something (clothes, etc.)" and "to hang somebody (word denoting human being, personal name, personal pronoun)". In the second case, "hang" will mean "kill" and translated accordingly.
2. The procedures used in our Machine Translation software programs can be used in all other related programs. These procedures can be used in the spell-checkers to provide corrections of meaning, when the sentence has no common sense and is not used allegorically, for example "John drunk the tree", our system "knows" that **tree** is not a **liquid** and not all liquids can be drunk by humans, etc.

IV) QUESTION ANSWERING: Our Question Answering software programs were converted several years ago into a Dialogue System (English, German chatbots), designed to provide plausible and accurate answers to direct questions or comment on non-interrogative text input. An accurate answer is provided when the answer is known for certain. For example, "Who is the president of the United States?" A plausible answer is provided when the exact answer is not

known. For example, "Where is Ann?" Since Ann is woman, most likely a housewife, probably she is at home. "Where is the pianist?" - the probable answer is "With the orchestra", etc.
The Dialogue System can remember user information, such as location of a key, medicament, money, etc. For example, if the user writes "*My blood pressure medicament is on the table*.", the System will remember where is the medicament. If the user forgets where is the blood pressure medicament, the user can ask the System "*Where is my blood pressure medicament*?". The System will retrieve the information. This ability of the chatbot can be extended to cover any valuable information.
In our Dialogue System, we have added the following type of word reference. For example: "*John attacked Susan, who was attacked*?", the answer will be "*Susan*". or alternatively, "John *attacked Susan, who is the attacker*?", the answer will be "*John*", etc.
Our Dialogue Question Answering System can be used in hand-held memory devices, able to provide true answers to people who forget or have many things to remember in daily life, as a better alternative to the date based diaries in the mobile devices. For example, "*When is my appointment with the dentist*?", etc.
One of the problems to resolve in such a dialog system is the Pronominal Reference (*I - my, you - your, John - he*, etc.). In the book "English Algorithmic Grammar" is provided an accurate algorithm for Pronominal Reference.

Date: 03.07.2023
LANGSOFT - Sprachlernmittel, SWITZERLAND