**Krzysztof Jassem** jassem@amu.edu.pl
Faculty of Mathematics and Computer Science, Adam Mickiewicz University, Poznań
**Mikołaj Wypych** wypych@amu.edu.pl
Faculty of Computer Science and Menagement
Poznan University of Technology, Poland
**Filip Graliński** filipg@amu.edu.pl
Faculty of Mathematics and Computer Science, Adam Mickiewicz University, Poznań

# Statistical and Heuristic Approach for Meaning Disambiguation in POLENG MT System

## *Abstract*

The paper addresses problems of ambiguities characteristic of Polish texts and shows some solutions applied in a Polish-English machine translation system, POLENG. First, a short description of the POLENG translation method is given. Then, the paper discusses typographic, lexical and structural ambiguities characteristic of Polish texts. Two ideas for resolving ambiguities are advanced. One of the concepts represents a theoretical approach – it enhances standard techniques used in probabilistic context-free grammars with the treatment of misspellings. A more practical approach is taken in the other concept. It applies a heuristic method for the scoring of parses. Finally, some examples are given, which show how the latter method may improve results of translation.

## *1. Introduction*

One of the major problems in a fully automatic machine translation using a transfer method is meaning disambiguation. Parsing tends to result in a set of admissible structures of a sentence (rather than one) because of language ambiguities. Two main types of ambiguities are usually discussed: lexical and structural. Another type of ambiguities is discussed here – one that results from typographic errors.

It is beneficial in terms of calculation complexity to disambiguate meanings before transfer process is activated. The question arises whether various types of ambiguities should be resolved by different methods or should they be treated by the same mechanism.

The paper advances two possible solutions. The first solution aims at building a language description that allows for the treatment of all types of ambiguities. The other solution concentrates on lexical and structural ambiguities only. Both solutions are motivated by the aim of translation from Polish into English. The latter solution has already been implemented in the POLENG system. The former one is currently being implemented. It will be interesting to compare the results of both methods.

## *2. POLENG translation method*

POLENG is a rule–based fully automatic machine translation system, which translates Polish texts into English[1]. The system has a modularized structure – particular stages of translation process are executed sequentially by separated modules. Linguistic data, interchanged between the modules, are stored in a specific structure, called ITF-list, which is universal enough to contain all types of information delivered by various modules.

The translation process is divided into the following stages: tokenization, sentence delimitation, lexical analysis, parsing, disambiguation, transfer and generation. Some of these stages are shortly characterized below.

[1] The research on the reverse direction of translation has been started recently

### 2.1. Lexical analysis

The objectives of lexical analysis are the following:

1) to recognize words that are included in the system dictionary
2) to recognize word chunks (groups of words which are treated by further modules as elementary units, e.g. *na pewno (English: for sure)*)
3) recognize lexical phrases described in the system dictionary
4) assign interpretations to unknown words.

In order to achieve the fourth objective a few techniques are applied: letter upcasing and downcasing (to distinguish between acronyms and regular words written in capitals), derivational analysis (see [Graliński]) and spelling correction. It is worth noting that obtained interpretations of unknown words are often ambiguous but the disambiguation task is left for later stages.

### 2.2. Syntactic-semantic parsing

The strategy of parsing in POLENG uses the CYK algorithm ([Aho, Ullman, 1972]). The formalism of grammar rules allows for the description of an attributive context-free grammar. A rule consists of four parts: production, operations on attributes, parse tree instructions, and scoring. Operations on attributes allow for unification and assignment of values to syntactic attributes as well as logical operations (such as intersection) on semantic attributes. Parse tree instructions make it possible to obtain a structural representation of a sentence so that it could form the input for transfer. Scoring is useful in the disambiguation algorithm discussed in Section 4.4.

### 2.3. Disambiguation

Disambiguation process may, in general, take place either before parsing (techniques of n–grams or transition networks may help resolve lexical ambiguities independent of structural information), during the process of parsing or after that process (a disambiguation algorithm may be designed that would evaluate all parses after they have been delivered by syntactic analysis, and that would select the best parse). In our approach, disambiguation takes place during parsing. The result of parsing is the set of parses – each of them being ranked by the or probability measures (see 4.3) or mechanism of scoring (see 4.4)

### 2.4. Transfer

Transfer converts one source structure – rated as the best parse – into a corresponding target structure. The rules of transfer are formalized in a language whose structure is similar to that of the C programming language. The language includes special instructions for operations on parse trees such as deletion and insertion of nodes and branches or assignation of values to attributes.

### 2.5. Generation

Generation is divided into three separate phases: syntactical modification, morphological generation and surface production. The first phase orders the components of the target sentence and modifies values of grammatical attributes, the second phase is responsible for the production of correct inflected and derived forms, the last stage produces the surface form. Generation uses the same syntax for the formulation of rules as transfer. All four types of rules (transfer, syntax, morphology, surface production) are interpreted and executed by the same mechanism.

## *3. Ambiguities*

This paragraph shortly sketches types of ambiguities characteristic of Polish texts.

### 3.1. Typographic ambiguity

Let $S$ denote a set of finite strings of characters. Let us define $G \subset S$ – a set of words recognized by a given lexicon – thereafter referred to as *dictionary*.

*A misspelling* is an ordered pair $(g, s) \in G \times S$, where $g$ denotes the string intended by the author of a text and $s$ denotes the string that actually occurred in the text instead of the intended one. There are two major types of misspellings:

*a)* $s \notin G$, e.g. (‘ambiguous’, ‘ambiguos’), (‘page’, ‘psge’).

*b)* $s \in G$: e.g. (‘through’, ’though’), (‘vary’, ‘very’).

Misspellings may arise as a result of typographic errors, data transmission, OCR errors etc. Although we distinguish a misspelling from an orthographic error (i.e. an erroneous word that resulted from the author’s poor knowledge of written language rather than a typo), the algorithms presented in Section 4.1, which aim at dealing with misspellings, may help handle some types of orthographic errors as well.

Misspellings result in ambiguities: whenever a fully automatic computer system comes across a string which is likely to be a misspelling (either a word from beyond the dictionary or a word that does not fit to an assumed structural frame), the system tries to guess the actual intention and usually ends up with a set of possible solutions.

An ambiguity, which appears as a result of a misspelling, is called *a typographic ambiguity*. Typographic ambiguities that are very characteristic of Polish texts are those connected with diacritic letters (ą, ć, ę, ś, ń, ź, ż). This is probably caused by the similarity between diacritic letters and non-diacritic letters (e.g. *ą* and *a*) and the standard techniques of inputting diacritic letters by way of key combinations.

### 3.2. Lexical ambiguities

Two types of lexical ambiguities may be distinguished for a highly inflectional language like Polish: caused by syncretism (different forms of the same lexeme are graphically identical) and caused by homonymy (forms of different lexemes are identical). The former type is not “dangerous” for translation: ambiguities are usually resolved by agreement rules and very rarely lead to ambiguity in translation.

Lexical ambiguities of the latter type are far more rare in Polish than in English. If they occur, however, they usually pertain to forms of absolutely unrelated lexemes. For example the form *liście* can be interpreted as:

- nominative and accusative plural of *liść* (Eng. *leaf*),
- dative and locative singular of *lista* (Eng. *list*),
- locative singular of *list* (Eng. *letter*).

There are 7292 homonymous words in the POLENG dictionary (1.8% of all words). If a homonymous word is misinterpreted during the analysis of a Polish text, then it will be mistranslated into English (and very often the equivalents of homonymous lexemes will have totally different meaning). Therefore, even though there are not many homonyms in Polish, it is crucial to avoid this type of mistakes.

It should be noted that some interpretations can be discarded *a priori* with a very small risk because they are highly improbable to occur in a real text. This concerns particularly vocative and imperative forms, e.g. *klucz* which is either a noun in nominative (Eng. *key*) or (unlikely) second person singular imperative of *kluczyć*, (Eng. *to weave the way*).

Obviously some misinterpretations are rejected during parsing because they do not fit any parse tree. More often, misrecognized words lead to a plausible but very unnatural parse: e.g. phrase *mam komputer,* meaning *I have a computer* could be, in theory, translated into English into *mums' computer*.

### 3.3 Structural ambiguities

There are various types of structural ambiguities in Polish. Among them attachment ambiguities are very common, especially PP-attachment ambiguities. Interestingly, quite often attachment ambiguities turn out to be "irrelevant" for Polish-to-English machine translation in a sense that even they are not resolved correctly, the final translation is acceptable. For example the English rendering of the sentence *Widziałem człowieka w sklepie* (*I could see a man in the shop*) simply shares the same ambiguity as the original. Similarly, resolving coordination ambiguities is not always vital in machine translation (e.g. *administratorzy i użytkownicy systemu* is as ambiguous as its English translation: *administrators and users of the system*).

In most cases, however, resolving structural ambiguities is essential for acceptable machine translation, particularly when equivalents of a phrase vary dependent on its subordinates. (e.g. the verb *ciążyć* is translated into *gravitate* if a PP with the preposition *ku* (Eng. *towards*) is attached to it, and *weigh down* otherwise.

## *4. Disambiguation*

In order to resolve ambiguities characterized in Paragraph 3. two techniques have been developed for the needs of the POLENG system. One of them assumes a statistical approach. The technique requires language modeling, which is described in sections 4.1 and 4.2. The technique itself is characterized Section 4.3. Section 4.4. presents a heuristic technique which has already been implemented in the POLENG system. Section 4.5 shows examples of applying this technique to selected Polish phrases.

### 4.1 Typographic ambiguity model

In the translation process performed by POLENG, misspellings are recognized at the stage of lexical analysis. The analysis, however, does not resolve typographic or lexicographic ambiguities. Instead, it produces probabilistic measures on ambiguous words. The measures are transferred to the stage of structural parsing. It is only then that the ambiguities are being resolved.

Such an approach allows for the resolution of typographic and lexical ambiguities with the aid of structural information.

The goal of typographic ambiguity model, for a given $s \in S$, is to return $C(s) \subset G \times [0; 1]$ such that for each $(g, q) \in C(s)$, $q = P(s \mid g)$ is the conditional probability of the misspelling $(g, s)$ and $q > q_0$ ($q_0$ is a certain assumed threshold). It is required that graphemes in $C(s)$ are unique and there is no other grapheme in $G$ matching the threshold criterion.

*Edit operation* is an atomic operation that modifies a string. We distinguish four edit operations:

1. *change* of single letter (including the null change), as in pairs: (‘election’, ‘electron’), (‘cat’, ‘cat’),
2. *deletion* of single letter, as in: (‘long’, ‘log’), (‘selection’, ‘election’),
3. *insertion* of single letter, as in: (‘phase’, ‘phrase’), (‘election’, ‘selection’),
4. *transposition* of two subsequent letters, as in: (‘terrain’, ‘terrian’), (‘on’, ‘no’).

For each misspelling ($g$, $s$) it is possible to define a sequence of edit operations – *edit chain* – which, applied to the string $g$, gives the string $s$. A trivial edit chain for a misspelling is the deletion of the source string and then insertion of all the characters of the target string. Of a special interest in text processing is the edit chain with the minimum value of a *cost function* for a given misspelling.

The classical approach defines the cost function as the sum of costs of edit operations that form a chain, where the cost of the edit operation as 0 for the null change and 1 otherwise [KUKISH]. The value of such a cost function is called Levenshtein distance. The algorithm for the calculation of C($s$) on the basis of Levenshtein distance was presented in [DACIUK].

Our approach uses *stochastic spelling correction* algorithm described in [WYPYCH]. In stochastic spelling correction we define the cost function as a negated probability of an edit chain (so that minimum cost would be realized by the most probable edit chain). The probability of an edit chain is calculated on the basis of a probability of edit operations comprising the chain. The calculation assumes that an edit operation is independent of all but the last preceding edit operations (then the edit chain may be treated as a 1st order Markov chain).

The most critical part of the algorithm is modeling of edit operations probabilities. The algorithm links a heuristic parameterized model with procedures of training. Training was based on Maximum Likelihood Estimation from a corpus of texts containing misspellings and their corrections.

The training corpus was obtained by gathering and correcting texts read by a lector and typed by 20 subjects. The total number of misspellings exceeded a thousand that allowed for a reliable estimation of parameters of the model.

So far, our experiments have concentrated on spelling mistakes that were defined in Section 3.1 as being of Type a). It is not clear to us at the moment, whether dealing with misspellings defined as Type b) is worth of the costs it would generate.

### 4.2 Lexical ambiguity model

Let $L$ denote the set of lexemes stored in the dictionary, let $P$ be a set of labels of parts-of-speech. Pair ($l$, $p$) is called *a word form* and $F=L\times P$ is the set of word forms. Let $s\in S$, $g\in G$, $l\in L$, $p\in P$ and $f=(l, p)\in F$. We expect the lexical analysis to provide information in the form of the set of $D(s)\subset G\times F$. In other words, for a given string $s$, information returned by lexical analysis consists of triples: grapheme ($g$), lexeme ($l$) and POS label ($p$). (If $s$ belongs to the dictionary, then $g$ equals $s$). We say that $s$ is typographically or lexically ambiguous iff $|D(s)|>1$.

We can formulate the goal of a lexical ambiguity model as to provide the assessment of P($s$, $f$), i.e. we expect the model to calculate the probability that a given string $s$ is meant to be a word form $f$ of a lexeme $l$.

The model suggested here assumes that for all $f\in F$ all $g\in G$, P($f$) and P($g$) are known (their values may be stored in the dictionary).

We expect our model to merely assess P($s, f$) (rather than give strict computation). Therefore we allow ourselves for some assumptions.

Notice that

$$\mathrm{P}(s,g,f)=\mathrm{P}(f\mid s,g)\mathrm{P}(s\mid g)\mathrm{P}(g). \tag{1}$$

P(*s, f, g*) is the probability that the string *s* was typed and the string was recognized as the grapheme *g* and the word form *f*. We assume that P($s \mid g$) is provided by the module of stochastic spelling correction. P(*g*) is known. We need to assess P($f \mid s,g$)

Now let us assume that

$$\mathrm{P}(f\mid s,g)\approx\mathrm{P}(f\mid g)=\mathrm{P}(l,p\mid g) \text{ and} \tag{2}$$

$$\mathrm{P}(p\mid l,g)\approx\mathrm{P}(p\mid l). \tag{3}$$

(2) assumes that the lexical probability distribution is independent of misspellings. Assumption (3) aims at simplifying calculations – it would be fairly complex to store all P($p \mid l,g$).

We go further on in with our approximations. We do not calculate all P($p \mid l$). We divide L into exclusive classes $L_i$ (for example $L_1$ may denote a class of noun lexemes which do not have plural forms), estimate an average $\mathrm{P}_i(p)$ for the whole class (on the basis of a corpus) and assume that $\mathrm{P}(p|l)=\mathrm{P}_i(p)$ iff $l \in L_i \subset L$,

Given that

$$\mathrm{P}(f\mid g)=\frac{\mathrm{P}(p,l,g)}{\mathrm{P}(g)}=\mathrm{P}(p\mid l,g)\mathrm{P}(l\mid g), \tag{4}$$

we obtain

$$\mathrm{P}(f\mid s,g)\approx\mathrm{P}(p\mid l)\mathrm{P}(l\mid g). \tag{5}$$

Substituting (5) into (1) we obtain the final form of approximation for P($s, g, f$).

Now, if we assume that $D(s)$ contains all graphemes that may have $\mathrm{P}(s, g, f)>0$ for a given *s* and any *f* we can calculate that:

$$\mathrm{P}(s,f)=\sum_{g\in\{g:(g,f)\in D(s)\}}P(s,g,f)\,. \tag{6}$$

### 4.3. Statistical approach to disambiguation

Information provided by models presented in 4.1 and 4.2 helps extend POLENG context-free grammar to a probabilistic context-grammar (abbreviated as PCFG, see e.g. [MW5]). PCFG is an extension of CFG which augments each rule P with a conditional probability: A $\rightarrow \beta$ [p] ([Jurafsky]).

An important part of the statistical approach is the estimation of probabilities of terminals in production rules. Robust systems tend not to store explicit rules for the production of terminals. The most specific rules produce parts-of-speech – possibly with constraints in attributive CFG's. This is also the case for POLENG. Such an approach poses problems for the calculation of the lowest branches of parse trees. The models described in 4.1 and 4.2 make it possible to estimate productions resulting in terminals.

Let $H \subset F$ be a set of word forms that may be derived from a given POS node. What we need is to estimate P($s \mid H$).

Let

$$F(s) = \{f : \exists_{g \in G}(g, f) \in D(s)\}.$$

Then

$$\mathrm{P}(s \mid H) = \frac{\sum_{f \in F(s) \cap H} \mathrm{P}(s, f)}{\sum_{f \in H} \mathrm{P}(f)}.$$

Typically, disambiguation techniques based on statistical approach compute probability of a parse tree on the basis of probability of production rules (see e.g. [Jurafsky]). There exist several improvements of this model based on extension of the information taken into consideration in assessing of probability of production (see e.g. [Collins]). The approach presented here extends those techniques with the statistical assessment of the terminal-ended branches of parse trees.

## 4.4. Heuristic approach to disambiguation

The model described in 4.3. is currently being implemented in the POLENG system. When the implementation has been completed, it will be possible to make comparisons to a heuristic approach, which is applied in a working system. The heuristic approach is sketched in this section.

Each parse of each segment in the input sentence is given *a score*. If the sentence has been successfully parsed as a whole, the parse with the highest score is selected and passed to the transfer engine. Otherwise – if only partial parses are returned – transfer is applied to the longest parsed segment starting from the left and the best-scored parse for the segment is selected (then the rest of the sentence is processed in the same way).

The score of a parse is calculated as the sum of scores of the subparses plus the extra contribution from the given node. The following four factors are taken into consideration in scoring of parses:

1. type of a parsed phrase – a sentence or a verb phrase is scored higher than a noun phrase, which in turn is scored higher than an adjectival phrase; incomplete phrases (e.g. a verb phrase without an obligatory complementation) are scored lower
2. number of words in a phrase which belong to lexical phrases included in the system dictionary – the more the better
3. scoring of the grammar rule which produced the parse (see section 2.2)
4. scoring of word forms according to their frequency in a corpus

The four factors are taken into consideration in the order specified above. For example, the scores induced by grammar rules are compared only when alternative interpretations represent phrases of the same type and the number of words belonging to lexical phrases is identical for both interpretations.

## 4.5. Examples of applying heuristic approach

The section presents a few examples of applying the heuristic approach described above.

*1) Mamy komputer*

This phrase can be interpreted either as a verb phrase (Eng. *we have a computer*) or as a noun phrase (Eng. *mum's computer*). The first interpretation is scored higher according to

criterion 1. Verb phrases are scored higher than noun phrases (except for titles, headers and lists).

*2) Komputer analizuje system.*

This sentence has two interpretations: having *komputer* (Eng. *computer*) as the subject and *system* as the object (*The computer analyses the system*) or the other way round (*The system analyses the computer*). The former one is more natural. POLENG grammar rules allow for parsing OVS sentences as well as SVO ones, but the latter ones are scored higher. *Komputer analizuje system* will be finally translated into *The computer analyses the system*.

*3) Aparatu fotograficznego Johna*

The desired rendering into English is:

- *of John's camera*

However, POLENG produces two other interpretations, which rendered in English would be:

- *of John's photographic apparatus*

- *of photographic John's apparatus* (the adjective is linked to a wrong noun because accidentally grammatical agreement rules cannot exclude it).

The first parse gets the highest score because it contains two words belonging to a lexical phrase (*aparatu fotograficznego - camera*).

*4) Listy w skrzynce*

This noun phrase can be interpreted either as a phrase in nominative where *listy* is interpreted as the plural nominative of *list*, Eng. *letter* (English translation of the phrase: *letters in the box*,) or as a phrase in genitive (English translation: *of a list in the box*), *listy* being interpreted as the singular genitive of *lista,* Eng. *list*). The first interpretation is selected because nominative forms of nouns are scored higher than other cases if no other clues are available.

## 5. Conclusions

The aim of the paper is to show that the Polish language poses new challenges for disambiguation in machine translation. This is caused by rich inflection and large likelihood of misspellings due to diacritic characters. The paper suggests two ways of facing those challenges. One solution enhances a traditional model for PCFG with the statistical estimation of probabilities of misspellings and probabilities of occurrences of word forms. The other method is heuristic – it reflects human expectations. The heuristic method has already been implemented in the POLENG system. The results of this method will form a basis for comparison with the statistical method.

## References


[Collins] Collins, M.J. (1999) *Head-driven Statistical Models for Natural Language Parsing*, Ph.D. thesis, University of Pennsylvania, Philadelphia.

[Daciuk] Daciuk J. (1998) *Incremental Construction of Finite-State Automata and Transducers, and their Use in Natural Language Processing*, Phd Thesis, Technical University of Gdansk, Poland

[Graliński], Graliński F., Krynicki G. *Word Formation Analysis in Polish-to-Eglish Machine Translation*, in: Speech and Language Technology, Vol. 7, Poznań, Poland. Available online at: www.ceti.pl/~poleng

[Jurafsky] Jurafsky D., Martin J. H. (2000), *Speech and language processing*, Prentice-Hall, New Jersey

[Kukish] Kukich K. (1992) *Techniques for Automatically Correcting Words in Text*, ACM Computing Surveys, Vol 24(4), pp. 377-439.
[Wypych] Wypych K. (2002) *Stochastic Spelling Correction of Texts in Polish*, in: Speech and Language Technology, Vol. 6, Poznań, Poland. Available online at: www.ceti.pl/~poleng