Text-Korpus からの検索について
—現代ドイツ語動詞の検索用正規表現の試み—

高瀬　誠

1．はじめに

近年、ドイツ語のテキスト型データベース（以下 Text-Korpus と記述する）がさまざまな形で作成・公開される様になってきており、ドイツ語・ドイツ文学研究にも有効に活用できるようになってきている。しかしながら、語形変化が比較的豊富な言語であるドイツ語のテキストから、目的とするデータを検索・収集するのは、今なおそれなりの手間がかかる作業となり、データの効率的な収集方法が大きな問題となる。本稿では、現代ドイツ語の Text-Korpus を利用した調査、特に動詞の検索の際に生じる問題点と、テキスト中でさまざまに変化している語形に対処する方法とを考察していきたい。

2．検索用文字列を考察する必要性

例えば et.[4] in Bewegung setzen, et.[4] außer Betrieb setzen, jn unter Druck setzen などのような、setzen を使う機能動詞結合にはどのようなものがあるのか調査する事を考えてみる。この場合、動詞 setzen をキーワードとして検索すれば、余分なものもヒットしてしまうと言う問題はあるものの、一応は必要としているものを残らず検索・抽出できる事になる[1]。し

1）ここでいう「余分なもの」の中には、機能動詞の用法ではない setzen の用例の他に、いわゆる分離動詞や非分離動詞が含まれてくる可能性もある。それらをここで目的としているものと区別する事は、時として難しい場合もある。3.2.2. 参照。

かし setzen は、setze, setzt, setzte, gesetzt 等々、比較的単純な変化をするとは言え、それなりの量の語形を持っている。

そうしたさまざまな形で出現する語句を Text-Korpus から検索・抽出するのに、あらゆるパターンを一つ一つ指定して何度も検索を行うのでは、いかに計算機の処理速度が向上しているとは言え、あまりにも非効率的であり、また操作ミスや不可抗力による検索洩れの確率が高くなる。キーワードになっている動詞(ここでは setzen)の変化形が全てヒットするような検索文字列を用意し、それを一度指定する事ができれば、作業は効率化し検索洩れなどの事故も防ぐ事ができるだろう。

3. 計算機による文字列検索の問題

3.1. Text-Korpus からの検索

一般に、欧文の Text-Korpus から一定の語を検索する際には、次のような点に留意する必要がある。

3.1.1. 大文字—小文字の区分による遺漏

例えば Sehen Sie dort gradeaus! と Wir sehen ihn im Park sitzen. という二つの文において動詞の形は、Sehen と sehen というように、見かけ上は語頭が大文字か小文字かの違いだけである。こうした場合については、最近利用されているどの検索ツールでも、きちんと対応できるようになっているのが普通であるが、場合によっては検索から漏れてしまう事もありうる。指定の仕方は利用するツールによってさまざまであるものの、問題としてはほぼ解決しているといってよい[2)]。

2) ただしドイツ語の場合は、3.1.7. と関連して同音異義語が混入してしまう可能性が残る。

3.1.2.　語形変化による遺漏

これは2で既に述べた通り、ドイツ語の様に変化形を豊富に持つ言語では特に問題となる。先に挙げた setzen の場合は比較的容易なケースだが、例えば強変化動詞 sehen などは sehe, siehst, sah, sähe, gesehen 等々、検索文字列を指定する事を考えるとはるかに煩雑になる。こうしたさまざまな語形変化に対応するには、柔軟なワイルドカード形式で検索文字列が記述できる必要がある。

3.1.3.　意図していない部分文字列の排除

例えば „echt“ という形容詞を検索しようとしたのにも関らず、„Berechtigung“ の下線部分にヒットしてしまった様な場合である。また、finden を検索する際に erfinden がヒットしてしまうという場合もある。この事は 3.1.7. にも関連するもので、検索文字列を注意深く設定することである程度は解決できるが、完全に除外するのは困難な場合もある。

3.1.4.　欧文特殊文字および記号の処理

これについては対象となる Text-Korpus のもつ基本構造に深く関っており、記述方式も Text-Korpus 毎にさまざまなのが実情で、一般化して言及する事は難しい。しかし、その記述方式によって検索文字列の指定も変ってくるので注意が必要である。

3.1.5.　複数行にわたる検索

これはしばしば問題となる点であるが、現行の検索用ツールの多くは、一行の中にある文字列を検索するのが普通である。必要とする部分が一行

に収まっている場合は問題ないが、そうでない場合は検索用文字列の指定や利用するツールを注意して選択する必要がある。

3.1.6. 単語の分綴

これも対象となるText-Korpusのもつ基本構造にも関連し、単語の分綴が行なわれているText-Korpusからの検索を細かく扱おうとすると問題が繁雑になってしまう。そこで本稿ではとりあえず、分綴が行なわれていないText-Korpusからの検索を行なうものとして、これを考察の対象から除外する事にする。

3.1.7. 同音異義語の混入

これについては他の項目に関する注意を払うことで、かなりのものを除外する事ができるが、完全に除外するのは極めて困難であり、問題を複雑化させる要因となり得る。例えば動詞gehenを検索する際に、Gehenという中性名詞にヒットしない様にするのはかなり難しい。そこでこのようなものがある程度混入する事を敢えて許容し、目的としている語句を全て検索・収集する事に努める方が、作業は効率的に行なえると思われる。

3.2. 現代ドイツ語動詞の検索

現代ドイツ語の動詞をText-Korpusから検索する際には3.1.で挙げた点と共に、特に以下の点に留意する必要があろう。

3.2.1. 変化パターンに合せた検索文字列を指定する必要性

言うまでもなく、ドイツ語の動詞は弱・強・混合・不規則といった変化パターンを持っている。これらに合わせる形で検索文字列を指定しなけれ

ばならない。特に、不規則な変化をする sein に関しては検索パターンは一つではなく、複数個指定する必要もあるだろう。

3.2.2. いわゆる分離動詞・非分離動詞と基礎語動詞の検索

geben, lernen 等のような接頭辞を持たない基礎語動詞を検索する際、注意しなければならないのは、abgeben や erlernen などのいわゆる分離動詞や非分離動詞の用いられた文にヒットしている可能性もある点である。いわゆる非分離動詞のケースは、ツールによっては単語の境界となっている文字列を指定する事もできるので[3]、これを利用する事によって非分離前綴のついた形を比較的うまく除外して検索が行えるが、いわゆる分離動詞の場合、それも主文で用いられている場合には、基礎語動詞単独で使用されている場合の例文との区別は極めてつきにくい。

もちろん、逆にいわゆる分離動詞の用例を検索する際には、この事を利用して検索を行うことができる[4]。その場合も同様に、基礎語動詞だけが使われているものとの区別は主文においてはつきにくい。特に3.1.5.と関連して、複数行にわたる文の場合には、かなり区別は難しくなる[5]。

3.1.3. で述べた事と関連して、基礎語動詞といわゆる非分離動詞の例が混在してヒットしてしまう可能性もある。例えば、stehen の過去分詞と gestehen の過去分詞などがこれにあたるであろう。

---

3) 例えば後述する正規表現がサポートされているツールであればこれが可能である。4.2.の表(2)を参照。
4) その際にはもちろんこれだけでなく従属文や過去分詞形の検索も必要になる。
5) 複数行にわたる検索文字列を扱えるツールも存在する。UNIX 系 OS 上のツールでは awk と Perl がそうしたものとして挙げられるが、この両者を利用する場合でも、検索文字列の指定に気をつけるのはもちろん、データの書式などがある程度整っていないと、徒にメモリを浪費する結果になる。

4. 「正規表現」について

現代ドイツ語のテキスト中から動詞を検索するのには前述のようにさまざまな壁がある。その壁を完全ではないものの、かなりの程度に克服できると考えられるのが「正規表現 (regular expression)」と呼ばれるものである。

4.1. 正規表現とは

正規表現は本来、UNIX 系 OS 上で動作する各種ツールで広く用いられている検索文字列の記述形式である。UNIX 系 OS の上で正規表現が利用できる代表的なツールには vi, mule (emacs), grep, sed, awk, Perl などがある[6]。他の計算機環境の上でも、例えばこれらの UNIX 系のツールが移植されたり、正規表現が利用できる様なその計算機環境に特有のツールが開発されたりして、正規表現はこれまで普及し、現在もなお普及しつつある。

4.2. 正規表現の特徴

正規表現の持つ最大の特徴は、普通の文字列に対してメタ・キャラクタを含める事で、極めて柔軟で強力なワイルドカード検索を可能にしている、という点であろう。つまり、„lernen“ や „lernt“ などのような固定した文字列を指定するばかりではなく、„lern“ という文字列の後に任意の文字列が続く、といった曖昧な検索も可能となるのである。

基本的なメタ・キャラクタとして概ね次のものが挙げられる:

6) もちろんこの他にも sh, csh, bash, zsh などの各種の shell をはじめ、 less, find など多数のツールで正規表現は利用できる。

メタ・キャラクタ	意味
^文字列	行頭
文字列$	行末
.	(改行文字以外の)任意の一文字
文字*	0回以上の繰り返し
[文字列]	[ ]内で指定したもののいずれか1文字
[a-z]	a から z までの範囲内のいずれか1文字
[^文字列]	文字列中に含まれない文字
\文字	文字そのもの(メタ・キャラクタの意味を打ち消す)

正規表現でのメタ・キャラクタ(1)

上記のメタ・キャラクタを利用するだけでもかなり柔軟なワイルドカード検索が可能となる。例えば、lernen の人称変化形は

[Ll]ern.*

と記述する事で、lern または Lern という文字列か、それぞれの直後に任意の文字列がついたものにヒットするので、過去分詞形以外のものを一応はカバーできてしまう[7]。

正規表現の記述方式は、ツール毎に拡張・変更が行なわれる事もあり、ある正規表現での記述がどのツールでも必ずしも使えるとは限らないが、ごく基本的な部分はどのツールでもほぼ共通している。今の所、もっとも拡張され強力なメタ・キャラクタを有しているツールは Perl であろう。以下に主として Perl で拡張されているメタ・キャラクタを挙げておく。

7) ただしこれでは „lernestn“ や „lernxxrt“ などの現実にはあり得ない文字列もヒットする対象になってしまうのでこの形をそのまま検索文字列に指定するのは問題が多い。

メタ・キャラクタ	意味	メタ・キャラクタ	意味
\b	単語の境界	\S	空白文字以外
\B	単語の境界以外	\d	数字
文字+	1回以上の繰り返し	\D	数字以外
文字?	0回または1回	\0	ヌル文字
{n,m}	n回以上m回以下の繰り返し	\ddd	8進数の表現
(…)	グループ	\xdd	16進数の表現
¦	選択	\cX	コントロール文字
\1,\2,…	後方参照	\n	改行文字
\10,…	後方参照(10番目以降)	\r	復帰文字
\w	単語の要素(英数字および,_')	\t	タブ文字
\W	単語の要素以外	\f	改ページ文字
\s	空白文字	\b	バックスペース文字

正規表現でのメタ・キャラクタ(2)[主として Perl での拡張]

こうした拡張されたメタ・キャラクタを利用する事により、より厳密な記述も可能となる。例えば前述の lernen の変化形に関して言うと

([Gg]e)*[Ll]ern[est]+n*d*

と記述する事により、全ての変化形をカバーする事が可能になる[8]。

4.3. 正規表現の持つ問題点

うまく利用できればかなり強力な記述方法となる正規表現にも問題となる点がある。正規表現はメタ・キャラクタを利用した柔軟なワイルドカードの指定が可能となっている反面、記述が繁雑になってしまい、一見した

8) しかしこれもまだ現実のドイツ語にはあり得ないパターンでもヒットしてしまう可能性が残ってしまう。

だけでは何を検索しようとしているのか分からなくなってしまう事もありがちである。

また、正規表現で検索文字列を指定しようする際、検索前からさまざまな可能性を考えるうちに、相当に複雑な検索文字列をいきなり指定してしまう事もある。こうした時に検索に失敗する事もしばしばある。

このような場合、いきなり複雑な正規表現を記述せずに、単純なパターンから出発して少しずつ記述を加える事で、検索対象を絞り込んでいく方がよいとされている。そのためにも自由に繰り返して検索できる環境を整えておくのが望ましい。

さらに前述の様に、正規表現はツール毎に細かい拡張が行なわれており、同じ事を行なう際にも、利用するツールによって異なった書き方をする必要が出てくる事もある。またあるツールではサポートされている機能が、別のツールにはない事もある。そのため検索内容に応じて利用するツールを的確に選択し、場合によってはそのために計算機環境を柔軟に変更できるようにしておく事も必要となろう[9]。

5. 現代ドイツ語の動詞の正規表現での記述について

正規表現を利用して、現代ドイツ語の動詞を Text-Korpus から検索する際には、どのような検索文字列を指定すればよいかをここで考察してみるが、その際に以下の点を前提とする。

1. 正規表現は Perl による拡張の行なわれたものを利用する。
2. 欧文特殊文字の記述方式は TEX のスタイルファイルである german.sty に従うものとする[10]。

9) 各ツールにおける正規表現のサポート状況に関しては、例えば [7],[14] などを参照。
10) 詳細については [5],[13],[20] を参照。

3. 3.2.2. で述べた様に、いわゆる分離動詞や非分離動詞の検索の問題があるが、ここでは問題を単純化するために基礎語動詞の検索に限定して取り扱う[11]。
4. 従って、検索に際しては目的としていないものが混入する可能性もあるが、遺漏はなくなるようにする。
5. 現実のドイツ語には存在しない語形にもヒットする可能性があるが、検索文字列の単純化のために、検索対象とする Text-Korpus にそのような語形は存在しないと仮定する。

また、正規表現である文字列群を記述しようとすると、その記述方法には、何通りかの可能性が出てくる事がしばしばである。例えば a, e, o のいずれかといった場合には [aeo] とも (a|e|o) とも記述できる。本稿では幾つかの記述の可能性のうちの一つを例示する事にする。

5.1. 弱変化動詞の検索

弱変化動詞の変化は、基本的には不定詞の語幹に一定の語尾をつけて行なわれる。これを文字列単位で見た場合に、どのようなヴァリエーションがあり得るのか概括しておく。例えば lernen の場合であれば以下のようになる:

lern*e*, lern*st*, lern*t*, lern*en*, lern*est*, lern*et*, lern*end*, lern*te*, lern*test*, lern*te*, lern*ten*, lern*tet*, *ge*lern*t*

これらの語尾の部分だけに着目すると、幾つかのグループに分類できる:

・-e の直後に何もつかないか n, st, t, nd がつくもの (-e, -en, -est, -et, -end)

11) 但し、ここで取り扱った検索文字列を実際に利用すると、いわゆる分離動詞や非分離動詞もヒットする事もあるのは、3.2.2. で述べたとおりである。

・t の直前に何もつかないか s がつくもの (-st, -t)

・-te の直後に何もつかないか t, st, n がつくもの (-te, -test, -tet, -ten)

さらにこれらを見ていくと、d, e, n, s, t の 5 文字の組み合わせである事がわかる。またこれらの一部にはある程度の順番がある。すなわち n と d が出現する場合、d は末尾のみ、n は末尾か d の直前である。このことを正規表現を用いるとn*d*と記述できる

また、e, s, t の 3 文字のいずれか 1 文字あるいはこの 3 文字の組み合わせが 1 回、この文字列の直前に出現する。ここまでを正規表現で記述すると [est]+n*d*となる。

これに不定詞の語幹部分を加えると lern[est]+n*d*となるが、この語幹部分の先頭は大文字となる可能性もある。そこでこれは [Ll]ern[est]+n*d* と記述する事になる。

また、過去分詞形は接頭辞 ge- をつけるが、これは他の人称変化形では出現しない。そこでこれをさらに加えると

([Gg]e)*[Ll]ern[est]+n*d*

となる。これで lernen のすべての変化形をヒットさせる事ができる。基本的にはこの語幹部分を変更する事で、他の弱変化動詞に応用する事が可能になる。

この正規表現を利用すると口調上の e が入る場合でも同じ記述方法がそのまま利用できる。例えば arbeiten の場合、

([Gg]e)*[Aa]rbeit[est]+n*d*

とすればよい。この場合も arbeite, arbeiten, arbeitest, arbeitet ばかりでなく、arbeitete, gearbeitet という語形にすべてヒットさせる事ができる。

ただし wandern などの動詞の場合、Wir wandern zusammen. などにヒッ

トさせるには語尾の部分を[est]*n*d*としなければならない。また、(ich) wandre という語形もあり得るので、最終的には

([Gg]e)*[Ww]ande?r[est]*n*d*

とする必要があろう。

5.2. 強変化動詞の検索

強変化動詞の場合も、語尾変化部分は弱変化動詞のものがほぼそのまま利用できる。ただし、[est]の直後は + ではなく、* となる。理由は直説法過去形で、ich sah などのようにこれらの文字列の組み合わせが出現しないパターンが存在するからである。

また、この動詞は語尾変化だけでなく幹母音も変化する。したがって強変化動詞の場合は弱変化動詞の場合とは違って、動詞の Ablautreihe に応じた語幹部分の記述を行う必要がある。

ここでは強変化動詞の Ablautreihe に従った検索文字列全てを列挙するのは控えるが、sehen を例に幹母音の変化パターンを見てみると次のようになる:

sehe, siehst, sah, sähe, gesehen

つまり e, ie, a, ä のいずれかである。この事を正規表現で記述すると ([ae]¦ie¦"a) となる。この部分を加えて sehen の変化パターンを正規表現で記述すると

([Gg]e)*[Ss]([ea]¦ie¦"a)h[est]*n*d*

となる。ここで問題となるのは、essen の様に幹母音だけではなく、語幹の子音も変化するケースである。essen の場合は幹母音が e, i, a, ä と変るばかりでなく、母音と語尾の変化に伴って子音も ss, ß のいずれかに変る。これを正規表現で記述すると ([ei]¦"?a)(ss¦"s) となる。また、essen は幹母

音で始まっている語なので、幹母音に大文字・小文字の区別をつける必要がある。そこでこれらをまとめると

([Gg]eg)*([eiEI]¦"?[Aa])(ss¦"s)[est]*n*d*

と指定することになる[12]。

5.3. 混合変化動詞の検索

上記二種類の動詞の検索パターンを踏まえれば、混合変化動詞の正規表現での検索パターンはさほど困難ではなかろう。例えば kennen の場合、語尾変化は弱変化動詞のものがそのまま利用できる。また、幹母音は kennen, kannte, kennte の様に変るので、[ea]とすればよい。すなわち、

([Gg]e)*[Kk][ea]nn[est]+n*d*

のようになる。

しかし例えば bringen の場合、bringen, brachte, brächte, gebracht というように語幹の子音が ng, ch と変化する。そこで

([Gg]e)*[Bb]r([ia]¦"a)(ng¦ch)[est]+n*d*

とすることで検索可能となる。

5.4. 不規則変化動詞の検索

例えば sein のような不規則な変化を起こす動詞についてみてみると、変化パターンは次のようなものに分類されよう:

- bin, bist, ist のタイプ
- sein, sind, seid, sei のタイプ
- war, wäre, gewesen のタイプ

12) もちろんこうした例は弱変化動詞にも見られる。例えば küssen などがこれに当たる。また、ß の表記規則は1998年8月を境に変更されるが、ここで挙げられている正規表現は新旧どちらの規則でも問題なく利用可能と思われる。[18] 参照。

これらをむりに一度に扱おうとしても検索パターンが徒に複雑になるばかりである。そこでこの動詞の場合はこれらのタイプ別に検索を行ったほうがよいと考えられる。

まず bin, bist, ist のタイプは

([Bb]i)(n¦st)¦[Ii]st

のように記述できよう。sein, sind, seid, sei のタイプのものは、幹母音が i, ei に変るほか、語尾が n, d などになる。そこで正規表現では、

[Ss](i¦ei)[est]*n*d*

の様に記述されよう。また、war, wäre, gewesen のタイプは、幹母音が a, ä, e のいずれかとなるほかに、語幹の子音も r, s と変る。また過去分詞形もこのタイプに属するので、ge- についての記述も加わる。以上をまとめて正規表現で記述すると、このタイプは

([Gg]e)*[Ww](es¦(a¦"a)r)[est]*n*

となろう。

6. 今後の課題

以上、正規表現を利用してドイツ語の基礎語動詞の検索パターンについて考察してきたが、ここに例示した以外の現代ドイツ語の動詞にも、正規表現による検索文字列を利用できるようにする必要があるなど課題も多い。

正規表現を利用しての検索には、それなりの基礎知識や技術力が必要となる事は否めない。もう少し簡単な検索を可能にするには、利用者に正規表現をあまり意識させなくてもよいユーザ・インターフェースも必要となろう。

また各種の Text-Korpus に対してネットワーク上からの検索も可能に

なってきている。正規表現による検索方法もネットワーク上から容易に行なえるようなものも必要となってくる。幸い Perl はネットワーク上での簡単なインターフェースを記述できるだけの機能も有しているので、今後はそうしたインターフェースと正規表現との連携を模索したい[13]。

13）ネットワーク上でのインターフェースを実現するのには、当該サイトのシステム管理者との協調が必要となるのは言うまでもない。

参考文献

[1] アスキー書籍編集部 編著:「MS-DOS を 256 倍使うための本 Vol. 1」, アスキー出版局, 1987.

[2]アスキー書籍編集部 編／福崎　俊博 著:「MS-DOS を 256 倍使うための本 Vol. 2」, アスキー出版局, 1987.

[3]アスキー書籍編集部 編／福崎　俊博・山田　伸一郎 共著:「MS-DOS を 256 倍使うための本 Vol. 3」, アスキー出版局, 1988.

[4]大木　敦雄 著:「入門 Mule」, アスキー出版局, 1994.

[5] 轡田　収／高瀬　誠:「OCR によるデータベース作成」学習院大学計算機センター年報 Vol. 16, 1995 所収.

[6] 小山　裕司／斎藤　靖／佐々木　宏／中込　知之 著:「UNIX 入門 — フリーソフトウェアによる最新 UNIX 環境」, 株式会社 トッパン, 1996.

[7] 小山　裕司／斎藤　靖／佐々木　宏／中込　知之 著:「Linux 入門」, 株式会社 トッパン, 1996.

[8] 坂本　文 著:「たのしい UNIX — UNIX への招待 —」, アスキー出版局, 1991.

[9]坂本　文 著:「続・たのしい UNIX — シェルへの招待 —」, アスキー出版局, 1993.

[10]志村　拓／鷲北　賢／西村　克信 共著:「AWK を 256 倍使うための本」, アスキー出版局, 1993.

[11] 砂原　秀樹／石井　秀治／植原　啓介／林　周志 共著:「プロフェッショナル BSD」, アスキー出版局, 1994.

[12] 砂原　秀樹／石井　秀治／植原　啓介／林　周志 共著:「プロフェッショナル・シェルプログラミング」, アスキー出版局, 1996.

[13] 高瀬　誠:「計算機による欧文テキスト処理について —機種依存し

ない統一的処理についての一考察—」学習院大学大学院ドイツ文学語学研究 第 18 号, 1994 所収.

[14]前田　薫／小山　裕司／斉藤　靖／布施　有人 共著:「Perl の国へようこそ」, サイエンス社, 1993.

[15]前田　薫／小山　裕司／斉藤　靖／布施　有人 共著:「新 Perl の国へようこそ」, サイエンス社, 1996.

[16] 松岡　裕典／松井　浩／網本　淳／原　信一郎 著:「MS-DOS テキストデータ料理学」, 翔泳社, 1992.

[17] Aho, A. V. / Kernighan, B. W. / Weinberger, P.J. 共著/ 足立　高徳 訳:「プログラミング言語 AWK」, 株式会社 トッパン, 1989.

[18] Heller, Klaus: Rechtschreibreform, Eine Zusammenfassung von Dr. Klaus Heller, in: SPRACHREPORT, Informationen und Meinungen zur deutschen Sprache, EXTRA AUSGABE Januar 1996.

[19] Lamb, Linda 著／福崎　俊博 訳:「vi 入門」, O'Reilly and Associates, Inc., アスキー出版局, 1992.

[20] Raichle, Bernd: „Kurzbeschreibung — german.sty (Version 2.5) 1. Januar 1995 (1. Korrektur 20. januar 1995)", in: emTEX Distribution von Version 19. Sep. 1995.

[21]Schwarz, Randal L. 著／近藤　嘉雪 訳:「初めての Perl」, O'Reilly and Associates, Inc., ソフトバンク株式会社, 1995.

[22]Wall, Larry / Randal L. Schwarz 著／近藤　嘉雪 訳:「Perl プログラミング」, O'Reilly and Associates, Inc., ソフトバンク株式会社, 1993.

Zum Suchen der deutschen Verben in Text-Korpora

— Ein Versuch der Zeichenkettenbeschreibung mit „Regular Expression“ —

Makoto Takase

Zur Zeit sind mehrere deutsche Texte elektronisch transformiert und als Text-Korpora veröffentlicht. Solche Texte, auf die man mit dem Computer Zugriff hat, sind für die germanistische Wissenschaft sehr nützlich (z. B. bei der Erforschung der Funktionsverbgefüge usw.). Aber beim Suchen eines Wortes aus einem Text-Korpus ergeben sich meistens technische Schwierigkeiten (z. B. Auslaufen der Daten, Berücksichtigung unerwünschter Zeichenketten und Homonyme usw.).

Mit „Regular Expression“, einer Beschreibungsweise für Suchzeichenkette, die hauptsächlich bei der Arbeit in UNIX-Umgebungen entwickelt wurde, könnte man die meisten Schwierigkeiten vermeiden. Die Beschreibung mit Hilfe der „Metazeichen“ ist der größte Vorteil der „Regular Expression“ und bietet eine Vielzahl von Möglichkeiten. Beispielsweise sind alle Flexionsformen der folgenden Verben so zu beschreiben[1]:

- lernen: ([Gg]e)*[Ll]ern[est]+n*d*
- essen: ([Gg]eg)*([eiEI]|"?[Aa])(ss|"s)[est]*n*d*

1) Hier wird vorausgesetzt, daß die nicht englischen Ziffern nach der TEX-Styledatei „german.sty“ definiert worden sind und daß in den Suchzeichenketten die von „Perl“ unterstützte „Regular Expression“ benutzt wird.

· bringen: ([Gg]e)*[Bb]r([ia]|"a)(ng|ch)[est]+n*d*

Für ein unregelmäsiges Verb wie „sein" müßte man mehrere Beschreibungen verwenden:

· ([Bb]i)(n|st)|[Ii]st

· [Ss](il|ei)[est]*n*d*

· ([Gg]e)*[Ww](es|(a|"a)r)[est]*n*

Mit diesen Beschreibungen könnte man alle Flexionsformen der betreffenden Verben in einem Text-Korpus suchen und als Forschungsmaterialien ausweisen. Es gibt aber auch Aufgaben, die noch nicht gelöst sind. Erstens muß man die Zeichenketten aller deutschen (mindestens der einfachen) Verben aufgrund der Ablautreihen beschreiben. Zweitens braucht man für das Verstehen und für die Benutzung der „Regular Expression" auch einige technische Kenntnisse. Es wäre außerdem nötig, ein einfacheres Benutzer-Interface bereitzustellen. Schließlich werden immer mehr nützliche Text-Korpora im „Network" veröffentlicht werden. Um sie auch wirkungsvoll nutzen zu können, braucht man ein Benutzer-Interface, das es, in Verbindung mit „Regular Expression", erlaubt, Suchzeichenketten einfach zu beschreiben.

(学習院大学非常勤講師)