A GENTLE INTRODUCTION TO REGULAR EXPRESSIONS (FOR USE IN SEARCHING LARGE CORPORA)[1]

John M. Dienhart and Henrik Kasch

ABOUT ISK'S LANGUAGE CORPORA

1. Language corpora available for ISK staff

Via the servers in our IT-Center, a number of very large language corpora are available to staff members at ISK. At the time of writing (July 2000) large text collections are available in the following five languages:

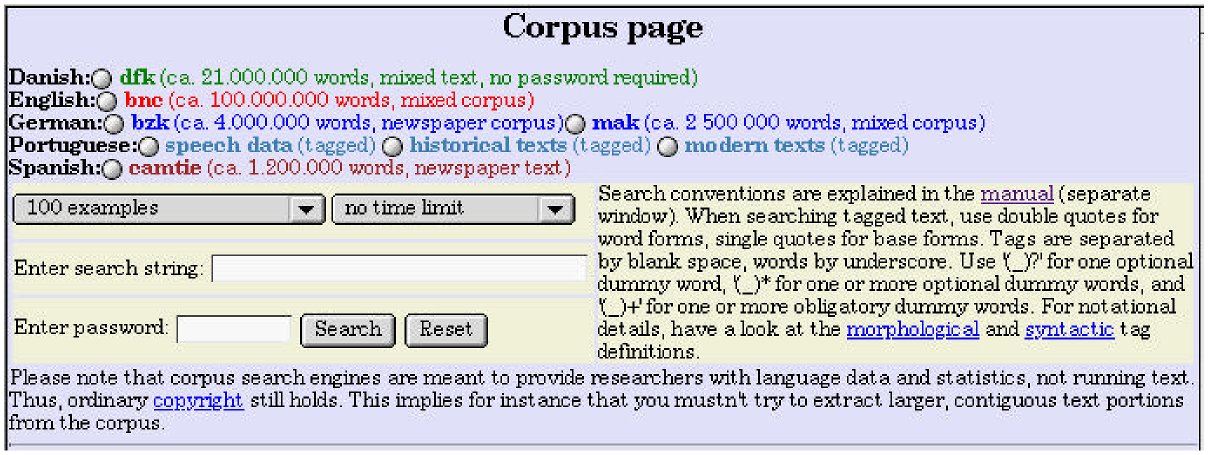
- **Danish** – The Danish corpus is nearing 21,000,000 words of mixed text. This corpus is being compiled by members of ISK and represents the first such initiative in Denmark. At the time of writing, the texts come primarily from two sources: political debates in the Danish Parliament (kindly provided by Parliament webmaster, Benny Høye), and an online news and literature magazine called *LOKE* (copyright Arne Herløv Petersen; *warning*: a number of the articles in *LOKE* are written in English). The editors of the archaeological journal *Skalk* have given us permission to scan in their back issues and add this material to the database. This is currently being done in the IT-Center. Also, Odense University Press is supplying us with electronic versions of books which they have published (after gaining permission from the authors). Consequently, the Danish corpus is growing steadily. *Acronym*: dfk (= 'dansk frikorpus'). *Password*: none required.
- **English** – This is the British National Corpus – consisting of approximately 100,000,000 words – an electronic copy of which has been purchased for use by ISK members. It is a mixed corpus covering a wide range of genres (both written and spoken). *Acronym*: bnc (= 'British National Corpus'). *Password*: required.
- **German** – The German texts are divided into two separately searchable corpora a) bzk (a newspaper corpus of approximately 4,000,000 words) and b) mak (a corpus of mixed genres, approximately 2,500,000 words). *Acronyms*: bzk (= 'Bonner Zeitungskorpus'), mak (= 'Mannheimer korpus'). *Password*: required.
- **Portuguese** – The Portuguese texts are divided into three separately searchable corpora: a) speech data, b) historical texts, and c) modern

[1] We would like to thank the following readers for helpful comments and suggestions: Eckhard Bick, Anders Hougaard, Kim Jensen, Rasmus Nielsen, Uwe Kjær Nissen and Anette Wulff.

texts. All the texts have been tagged with the VISL-tools in co-operation with the following research teams: a) the CORDIAL-SIN project (dialect speech data), b) the TYCHO BRAHE Corpus of Historical Portuguese (historical), and c) the NILC project (modern written Brazilian Portuguese). *Password*: required.
- **Spanish** – The Spanish corpus consists of approximately 1,200,000 words from two news magazines. *Acronym*: camtie. *Password*: required.

2. Gaining access to the language corpora

Using Netscape Communicator (Navigator) or Internet Explorer you can go directly to ISK's corpus server (http://corp.hum.sdu.dk), where you will find the following 'Corpus page' designed and implemented by Eckhard Bick:



Corpus page

Danish: ○ **dfk** (ca. 21.000.000 words, mixed text, no password required)
English: ○ **bnc** (ca. 100.000.000 words, mixed corpus)
German: ○ **bzk** (ca. 4.000.000 words, newspaper corpus) ○ **mak** (ca. 2 500 000 words, mixed corpus)
Portuguese: ○ **speech data** (tagged) ○ **historical texts** (tagged) ○ **modern texts** (tagged)
Spanish: ○ **camtie** (ca. 1.200.000 words, newspaper text)

100 examples ▾ no time limit ▾

Enter search string:

Enter password: Search Reset

Search conventions are explained in the manual (separate window). When searching tagged text, use double quotes for word forms, single quotes for base forms. Tags are separated by blank space, words by underscore. Use '(_)?' for one optional dummy word, '(_)*' for one or more optional dummy words, and '(_)+' for one or more obligatory dummy words. For notational details, have a look at the morphological and syntactic tag definitions.

Please note that corpus search engines are meant to provide researchers with language data and statistics, not running text. Thus, ordinary copyright still holds. This implies for instance that you mustn't try to extract larger, contiguous text portions from the corpus.

Fig. 1

2.1 Setting the search parameters

You will note in Figure 1 that there are several parameters involved in any given search. Here is a description of each of the parameters:

- Start by selecting the appropriate corpus for your search. You do this by clicking in the circle to the left of the corpus you wish to use. Note that there are two choices for German and three for Portuguese. You may click in only ONE circle for any given search.
- Select the number of examples you wish to find. Note that this is automatically set to 100 (such an automatic setting is known as a default condition), but you can change this by clicking on the scroll bar in the relevant menu. When you do this, you will note that the current options are: *1, 10, 100, 1000, all, just count*. If you select 'all', you risk filling memory. If you select 'just count', you will be told

how many instances of the given string appear in the selected corpus, but no examples will be listed.

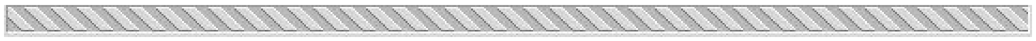
- Select the appropriate time limit. There are currently two time options: *10-20 seconds*, and *no time limit*. Note that the default condition is 'no time limit', but you can change this by clicking on the scroll bar in the relevant menu.
- Enter a search string. (In sections 4-7 below, you will be given examples of many different types of search strings – all of them in the form of so-called 'regular expressions'.)
- Type in a password. At the time of writing, a password is needed for all the corpora except the Danish one. You can get a password by contacting the head of the IT-Center (currently John Dienhart).
- Click on 'Search' to start the search for the specified string.

Note: The 'Reset' button can be used if you wish to reset all the parameters to their default conditions (that is: no corpus selection, 100 examples, no time limit, no search string, no password – as in Figure 1).

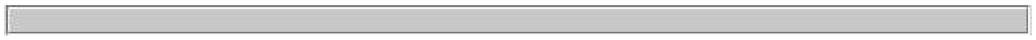
2.2 Some useful notes

Note 1: We recommend that you select *10 examples* and *10-20 seconds*, while you are experimenting with the design of appropriate search strings. This is because you may make one or more errors in preparing for your search, and it is therefore desirable to have such a search terminate quickly.

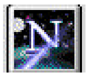
Note 2: Since it often takes several seconds, sometimes minutes, to search through a large corpus, you should be aware of the clues your browser provides to let you know that the search process is on-going. In Netscape, for example, the time bar at the bottom of the screen will look like this if the search is on:



and like this if the search is over (or never got started because you forgot to specify one of the parameters):

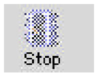


Another clue provided by Netscape is visible motion (comets and such) inside the Netscape icon if the search is in progress:



There will be no motion inside this icon if no search is in progress. Regardless of how many examples you ask for, the search engine is currently designed to supply you with a count of the total number of occurrences, after it supplies the specified number of examples. This means that the search will continue even after the list of matches has appeared on your screen.

To stop the search at any time, click (twice) on the 'Stop' button in your browser. In Netscape the icon looks like this:



Note 3: If you fail to get any results from a given search, this could be due to one or more of the following reasons:

- You have forgotten to click in a button next to the desired corpus.
- You have forgotten to supply the password.
- You have supplied an incorrect password.
- You have forgotten to specify a search string.
- The string you have specified does not appear in the selected corpus (for example, instead of pressing the space bar, you may have used _, a character which is relatively rare in texts; or perhaps you accidently put in two spaces at the beginning or end of the search string when you intended only one).

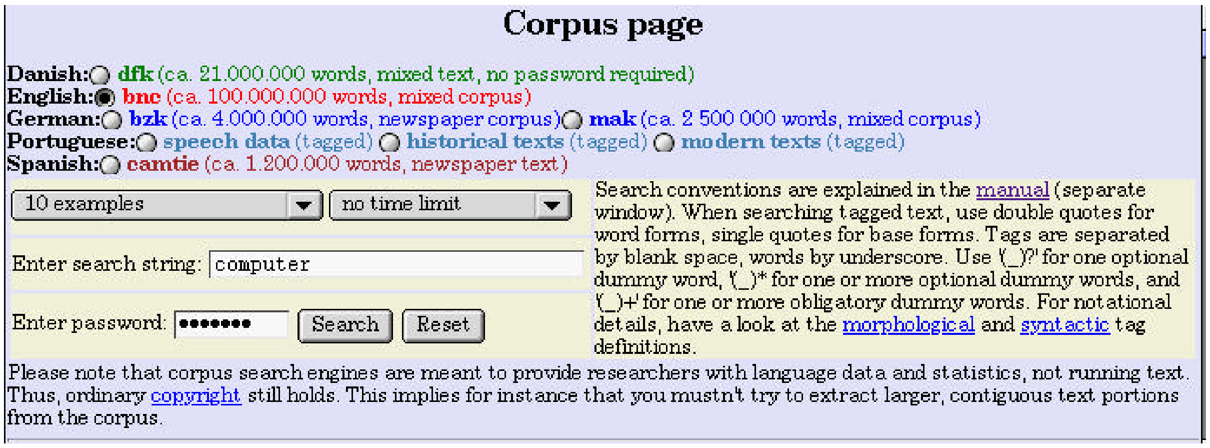
Make the appropriate adjustments and try again.

3. Eckhard Bick's search engine on the corpus server

The server on which the language corpora are stored runs under a Linux system, and Eckhard Bick has created a search engine which supports regular expressions (the subject of sections 4-7 below) – thus greatly enhancing the power of the search engine. The regular expressions are interpreted automatically by a Linux tool known as 'grep'. The grep command is executed when the search is initiated. Grep-based searches can be used to retrieve any text string by matching the string in question against corpus data. Whenever a text string is matched, Eckhard's search engine supplies the search results in the form of a concordance with the keyword (the string searched for) in its proper context and **boldfaced**. Each search result is also supplied with id-buttons which can be activated in order to provide more context for the keyword in question.

3.1 A sample search result

Let us consider a straightforward illustration of the features described above by searching for the first 10 occurrences of the string, *computer*, in the BNC: After we have set the appropriate parameters, the 'Corpus page' will have the form displayed in Figure 2:



Corpus page

Danish: ○ **dfk** (ca. 21.000.000 words, mixed text, no password required)
English: ● **bnc** (ca. 100.000.000 words, mixed corpus)
German: ○ **bzk** (ca. 4.000.000 words, newspaper corpus) ○ **mak** (ca. 2 500 000 words, mixed corpus)
Portuguese: ○ **speech data** (tagged) ○ **historical texts** (tagged) ○ **modern texts** (tagged)
Spanish: ○ **camtie** (ca. 1.200.000 words, newspaper text)

10 examples | no time limit

Enter search string: computer

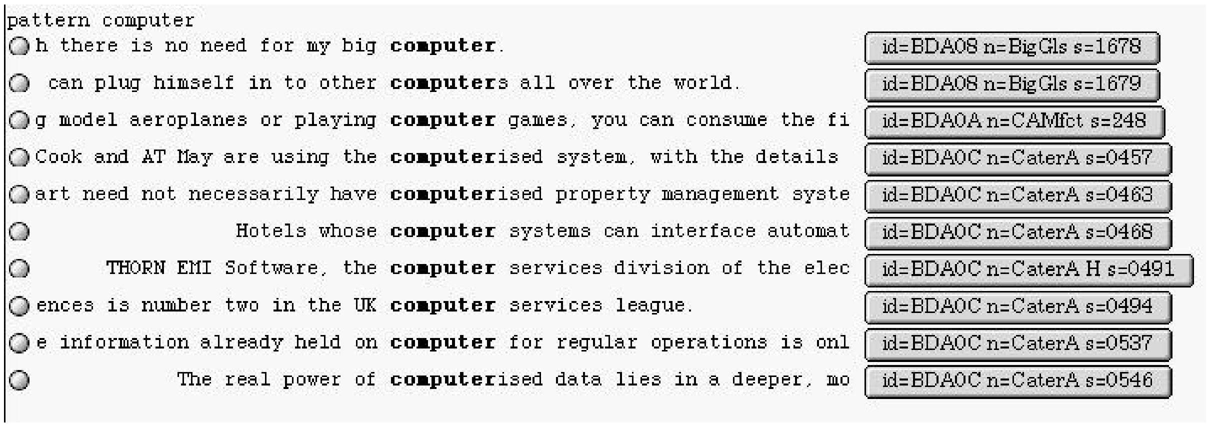
Enter password: ●●●●●●● [Search] [Reset]

Search conventions are explained in the manual (separate window). When searching tagged text, use double quotes for word forms, single quotes for base forms. Tags are separated by blank space, words by underscore. Use '(_)?' for one optional dummy word, '(_)*' for one or more optional dummy words, and '(_)+' for one or more obligatory dummy words. For notational details, have a look at the morphological and syntactic tag definitions.

Please note that corpus search engines are meant to provide researchers with language data and statistics, not running text. Thus, ordinary copyright still holds. This implies for instance that you mustn't try to extract larger, contiguous text portions from the corpus.

Fig. 2

(Observe that the password has automatically been concealed by the use of asterisks.) To start the search, we now click on the 'Search' button. The result is shown in Figure 3:



```
pattern computer
○ h there is no need for my big computer.                                 id=BDA08 n=BigGls s=1678
○  can plug himself in to other computers all over the world.             id=BDA08 n=BigGls s=1679
○ g model aeroplanes or playing computer games, you can consume the fi    id=BDA0A n=CAMfct s=248
○ Cook and AT May are using the computerised system, with the details     id=BDA0C n=CaterA s=0457
○ art need not necessarily have computerised property management syste    id=BDA0C n=CaterA s=0463
○                  Hotels whose computer systems can interface automat    id=BDA0C n=CaterA s=0468
○       THORN EMI Software, the computer services division of the elec    id=BDA0C n=CaterA H s=0491
○ ences is number two in the UK computer services league.                 id=BDA0C n=CaterA s=0494
○ e information already held on computer for regular operations is onl    id=BDA0C n=CaterA s=0537
○                The real power of computerised data lies in a deeper, mo  id=BDA0C n=CaterA s=0546
```

Fig. 3

Suppose that you wish to see the complete sentence for the fourth match displayed in Figure 3. Simply click in the circle to the left of the example. This will supply the following context in the 'Context window' at the bottom of the screen:

> id=BDA0C n=CaterA s=0457 Selected branches of travel agents Thomas Cook and AT May are using the **computerised** system, with the details of about 100 selected hotels.

The tag at the beginning of the sentence identifies the source and location of the sentence in the BNC. This tag is always visible in the larger 'button' to the right of each string in the search results (see Figure 3). If you need an even larger context, you can click on this button. Doing so for the current example provides the following context for the sentence in question (note that now the sentence itself is boldfaced):

> SOLUTIONS INDEPENDENTS GET BREAK FROM BRAVO by David Goymour BRAVO, a new on-screen booking system which puts British hotels and tourist attractions on travel agents' counters, has entered its launch phase. It is said to offer independent hotels the kind of exposure which hotels in big groups can derive from international booking systems &mdash owned, typically, by the big airlines. **Selected branches of travel agents Thomas Cook and AT May are using the computerised system, with the details of about 100 selected hotels**. Bravo is planning to have 1,000 hotels on the system by 1 November, and 2,000 by March 1992. Majority shareholder in Bravo is Cable & Wireless (C&W), which also owns Mercury, the communications company. C&W's partners are British Airways and Finite, a technology specialist. &bquoBravo will distribute &bquoUK Ltd&equo worldwide, and give the travelling public access to the total UK product,&equo

(*Note*: The careful reader may have spotted the curious labels *&mdash*, *&bquo*, and *&equo* in the above text. These are HTML codes (for, respectively, 'dash', 'begin quote', and 'end quote') which for one reason or another have not been filtered from the text. Note, too, that because tags such as *id=BDA0C n=CaterA s=0457* are present (but invisible) in the corpus itself, they can appear inadvertently in search results – try searching for the string *id*, for example!)

3.2 Starting a new search

If your computer screen currently displays a series of search results and you wish to return to the 'Corpus page' to start a new search, there are two basic options.

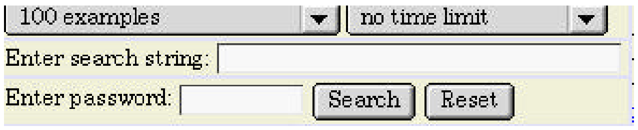
- *Option 1*: click on the 'Back' button in your browser:

  This will return you to the 'Corpus page' while keeping all the current parameters (that is, the selected corpus, number of requested matches, time limit, search string, and password).

- *Option 2*: Alternatively, you can follow the instruction in the search engine at the top of the search result page, where it says: *For a new* *search click here!* This, too, will return you to the 'Corpus page', but all parameters on this page will be reset to their original default conditions (you will then have to choose your corpus again and reenter both the search string and the password).

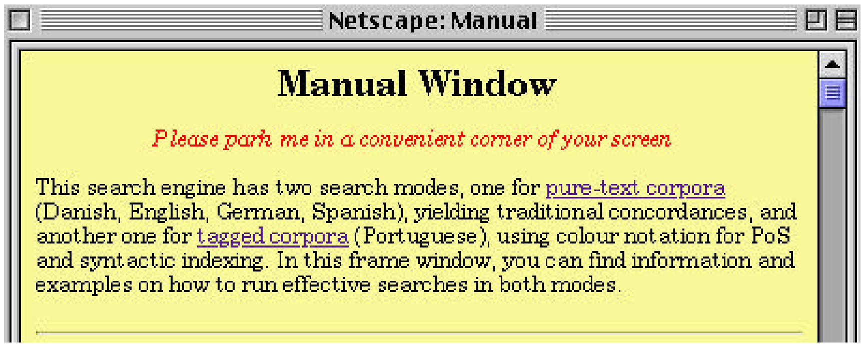
We recommend that you use the 'Back' button, since then you will not have to specify the desired corpus again (presumably you will be working on the same corpus for a while), nor will you need to retype the password.

Should you wish to redo all the parameters, simply click on the 'Reset' button:



3.4 Eckhard Bick's 'Manual Window'

The rest of this pamphlet illustrates a variety of relatively straightforward regular expressions, and lists some of the results yielded when searching for these patterns in the different corpora. For convenience, Eckhard Bick's search site contains a 'Manual Window' (see Figure 4) which provides a succinct summary of many of the basic symbols used in regular expressions.



Manual Window

Please park me in a convenient corner of your screen

This search engine has two search modes, one for pure-text corpora (Danish, English, German, Spanish), yielding traditional concordances, and another one for tagged corpora (Portuguese), using colour notation for PoS and syntactic indexing. In this frame window, you can find information and examples on how to run effective searches in both modes.

Fig. 4

This 'Manual' pops up every time you return to the 'Corpus page. It will disappear automatically when you start the search (that is, when you click on 'Search'), but will reappear when you start a new search. To remove it (temporarily), click on the little square in the upper left-hand corner of the 'Manual Window', or follow the instructions in the second line of the window itself: 'Please park me in a convenient corner of your screen.' You can 'park' the window by dragging it to some new location (use the mouse to 'grab' the window at the top and then drag).

REGULAR EXPRESSIONS

4. About regular expressions

A 'regular expression' is a technical term referring to a specially constructed string of one or more characters which can be used for pattern matching. Consequently, regular expressions can be very useful in searching for various types of data in language corpora – provided that the search engine you are using supports regular expressions. As noted above, Eckhard Bick's search engine supports these expressions.

A regular expression consists of one or more 'standard' characters (e.g. a, R, 9, etc.) combined with zero or more 'special' characters (e.g. *, ^, ?, etc.). The special characters (which will be defined and illustrated below) each have a special function, and hence they cannot be used as 'standard' characters without employing a special technique. (This technique involves the use of the backslash (\) – but more on this below.)

We shall start by describing regular expressions which contain only standard characters (section 5), then move on to an examination of the function of individual special characters (section 6). Next we turn to examples of regular e xpressions containing standard characters combined with two or more special characters (section 7) and conclude with a set of exercises (section 8) which you may wish to try to solve, using what you have learned about regular expressions. Appendix 1 contains a summary of all the regular expressions presented in sections 5, 6, and 7, while Appendix 2 provides suggested solutions to the exercises which are given in section 8.

Illustrative examples are taken from the following corpora: bnc (E = English), dfk (D = Danish), mak (G = German), camtie (S = Spanish). The basic structure of each example is as follows:

- *Goal*: a description of the desired search
- *Regular expression*: an example of a regular expression meeting the goal
- *Result*: a list of strings from the selected corpus which match the pattern described by the regular expression.
- *Comments* and, occasionally, *Questions* to ponder

Note: Since a space (or blank) is a standard character, and a very useful one at that, it is often found in regular expressions. However, a space can be difficult for the reader to spot, especially when it occurs at the beginning or end of a regular expression. Consequently we have used the symbol _ to represent space. Note that _ is NOT A CHARACTER WHICH YOU TYPE IN, but simply a means of instructing you to press the space bar.

5. Standard characters only

- **Goal**: find words containing the string *ess*
  Regular expression: ess

Result:

D	∧	*interesseret, fællesskab, adresser, presse,* etc.
E	∧	*Professor, unless, unnecessarily, access, expressed,* etc.
G	∧	*besser, fressend, Professor, Obstmesser,* etc.

Comment: In this very simple regular expression, only standard characters are employed. There are no special characters in the string. The pattern will match any string containing *ess*.

- **Goal**: find words ending in *ess*
  Regular expression: `ess_`
  Result:

E	∧	*unless, access, stress, press, success,* etc.
G	∧	*Prozess, Kongress, Stewardess*

  Comment: Here again, no special characters have been used, but by putting a space at the end of the string, we ensure that our results contain only words ending in *ess* (recall that _ is not a character which you type in, but simply a means of instructing you to press the space bar; hence what you actually type is 'ess ' – ignoring the quotation marks, of course). Note that the above regular expression will find words ending in *ess* only if these words are followed by a space. A match will not occur if, for example, the word is followed by a punctuation mark instead of a space (thus the word *blindness* will not be found in a string such as 'sudden blindness, dementia, dramatic weight loss'). We will see shortly how regular expressions can be designed to deal with this issue.

- **Goal**: find words starting with *scr*
  Regular expression: `_scr`
  Result: E ∧ *screened, scrutiny, scrap, script, scrupulous,* etc.

  Comment. Without the leading space, this search would have returned (in addition to the above matches) such strings as *prescription, described,* and *transcripts*.

  Question: No Danish, German, or Spanish examples of this consonant sequence occur. Why not? What change(s) must be made in the regular expression to yield 'equivalent' results in Danish? German? Spanish?

- **Goal**: find instances of the word *low*
  Regular expression: `_low_`
  Result: E ∧ *low*

Comment: Without the leading and trailing spaces (_), this search would have returned (in addition to *low*) such strings as *below, allowing, follow,* and *slowly.*

6. Individual special characters

This section introduces several of the most useful special characters and shows how they can combine with standard characters to create patterns which go beyond the simple string of standard characters, thereby increasing the power and flexibility of the search.

6.1 The full stop (.)

This special character stands for any character and exactly one. Here are some examples:

- **Goal**: find five-letter strings that end in *ing*
  Regular expression: `_..ing_`
  Result:

  D ∧ *sving, iling, uting, tving,* etc.

  E ∧ *being, dying, going, thing, bring, cling,* etc.

  G ∧ *bring*

- **Goal**: find four-letter strings which start and end in *b*
  Regular expression: `_b..b_`
  Result: E ∧ *bomb, bulb, blob,* etc.

  Question: Can you predict what type of results you would get (from the BNC) if you left off the trailing space? Try it and see.

6.2 The question mark (?)

This special character stands for no instance or one instance of a preceding character (or a preceding expression, which must be in parentheses – see section 7.2).

- **Goal**: find instances of the singular and the plural forms of *dog*
  Regular expression: `_dogs?_`
  Result: E ∧ *dog, dogs*

- **Goal**: find instances of *nice* with or without quotes
  Regular expression: `_"?nice"?_`
  Result: E ∧ *nice, "nice", "nice, nice"*

  Comment: Since *?* means no instance or one instance of the preceding character, *"?* means 'with or without a quotation mark'. Consequently, the above regular expression will find occurrences of *nice* whether or not they are surrounded by quotation marks (of the type ").

6.3 The asterisk (*)

This special character stands for any number of instances (including none) of a preceding character (or a preceding expression, which must be in parentheses – see section 7.2).

- **Goal**: Find words ending in *stle* or *sle*
  Regular expression: `st*le_`
  Result:

D	∧	*risle, udveksle, varsle, usle,* etc.
E	∧	*castle, apostle, isle, Carlisle,* etc.
G	∧	*wechsle, fröstle*

6.4 The plus sign (+)

This special character stands for one or more instances of a preceding character (or a preceding expression, which must be in parentheses – see section 7.2).

- **Goal**: Find words containing *eting* or *etting*
  Regular expression: `et+ing_`
  Result:

D	∧	*Folketing*
E	∧	*getting, meeting, letting, marketing,* etc.

  Comment: Observe that while *–eting* and *–etting* are quite common endings in English words, they is rare in Danish (but cf. proper nouns such as *Stetting*). Observe, too, that this regular expression would also find instances of words ending in *ettting, etttting,* etc. if there were any.

6.5 The vertical bar (|)

This symbol separates expressions, allowing a search for alternative patterns. (Alternatives are surrounded by parentheses if they are part of a larger expression; see section 7.2.)

Note: If you cannot find the vertical bar on your keyboard, try <alt><i> (for Mac users) or <Alt Gr><´> (for PC users).

- **Goal**: Find words containing either *ee* or *oo*
  Regular expression: `ee|oo`
  Result:

D	∧	*ideer, reelle, sneen, voodoo, udseende,* etc.
E	∧	*blood, between, schools, need,* etc.
G	∧	*leer, Kaffee, doofen, sooft, Schnee,* etc.
S	∧	*cree, deseemos, cooperación, leer, coordine,* etc.

- **Goal**: Find instances of forms of the irregular verb *go*
  Regular expression: `_go_|_goes_|_gone_|_going_|_went_`
  Result: E ∧ *go, goes, gone, going, went*

  Comment: See section 7.2 for an alternative search string yielding the same results.

6.6 Square brackets ([])

Square brackets are used to surround a set of characters used in a search. A match on any one of the characters in the set will count as successful. Thus [aeiou] is the set of (lowercase) vowels (excluding <y>). A hyphen can be used to abbreviate some sets. Thus [0-9] is equivalent to [0123456789], and [ABCDEFGHIJKLMNOPQRSTUVWXYZ] can be shortened to [A-Z]. Smaller sets can, of course, be defined, such as [a-d] and [1-5].

Note: If you have trouble finding square brackets ([and]), try <alt><8> (for [) and <alt><9> (for]).

- **Goal**: Find English words ending with a vowel
  Regular expression: `[aeiouy]_`
  Result: E ∧ *immune, a, the, to, so, defence, very, society,* etc.
- **Goal**: Find words starting with *s* and followed by *p, t,* or *c* (that is, strings starting with *sp, st* or *sc*)
  Regular expression: `_s[ptc]`
  Result:

D	∧	*sparer, stoffet, scanner,* etc.
E	∧	*still, school, speak,* etc.
G	∧	*schon, steigt, später,* etc.

- **Goal**: Find words starting with a capital letter and whose second letter is not a capital letter
  Regular expression: `_[A-Z][a-z]`
  Result:

D	∧	*Det, Ved, Beholde,* etc.
E	∧	*Immune, This, How,* etc.
G	∧	*Ansichten, Heinrich, Roman,* etc.

S	∧	*Esta, Para, Nueva,* etc.

Comment: Such a search string can be useful for finding the beginnings of sentences, proper nouns, or German nouns in general.

Question: What would be the result of the above search if the second set ([a-z]) were omitted in the regular expression? Try it and see.

- **Goal**: Find words starting with *Aa* or *aa*
  Regular expression: `_[Aa]a`
  Result:

D	∧	*Aarhus, Aalborg, aabned,* etc.
E	∧	*Aaron, aah, Aachen,* etc
G	∧	*Aasgeier, Aale, aalvimmelnden,* etc.

- **Goal**: Find two-digit numbers
  Regular expression: `_[0-9][0-9]_`
  Result: *10, 40, 24,* etc.

6.7 The backslash (\)

The backslash is used to change the status of the character immediately following it. It can thus be used to change a 'special' character into a 'standard' character, (e.g. \.), or to change a 'standard' character into a 'special' character (e.g. \w). (The character after the backslash is referred to as an 'escape character'.)

Note: if the backslash is not on your keyboard, try the combination <shift><alt></>).

Here are some examples.

Example 1: \.

By preceding the full stop with a backslash, the special status of the full stop ('any character and exactly one') is overridden, and the full stop becomes a simple mark of punctuation.

- **Goal**: Find instances of the form *etc.*
  Regular expression: `etc\.`
  Result: *etc.*

  Comment: If one searched for *etc.* instead of *etc*\. the full stop would take on its special function and the result would be such strings as: *etc., stretched, sketches,* and *Fletcher*.

- **Goal**: Find instances of the form *m.m.* in Danish
  Regular expression: `m\.m\.`
  Result: D ∧ *m.m.*

  Comment: Remember to select the Danish corpus before you start the search.

Example 2: \w

It is often useful to be able to search for what are known as alphanumeric characters – that is, letters of the alphabet or numbers, excluding all marks of punctuation. This can be done by means of the set [a-zA-Z0-9]. Alternatively, one can use *w* as a simple shorthand for this set.

- **Goal**: Find words containing *st*, then any character, then *d*
  Regular expression: `st\wd`
  Result:

  D ∧ *sted, vækstideologi, forstod,* etc.

  E ∧ *listed, studied, tested, custody,* etc.

G ∧ *student, Heimatstadt, Seemannstod,* etc.

S ∧ *Usted, atestado, Estados,* etc.

Question: At first glance, one might think that the regular expression *st.d* would yield the same results as the one above. However, this is not the case. Why not? Try it and see.

- **Goal**: Find English compounds of the type *willy-nilly,* where each member of the compound is 5 letters long and ends in *y*
  Regular expression: `_\w\w\w\wy-\w\w\w\wy_`
  Result: E ∧ *fuddy-duddy, willy-nilly, hurly-burly, hanky-panky,* etc.

  Comment: Though this finds primarily rhyming compounds, it also yields strings such as *fifty-fifty, buddy-buddy,* and *topsy-turvy.*

7. Combinations of special characters

Special characters are combinable. This provides even more powerful patterns.

7.1 Negation (^)

The caret (^) can be used inside square brackets in order to negate one or more members of the set.

Note: Generally, the caret will not appear on the screen until you press the space bar after typing the caret itself. This is because the caret is a diacritic which typically appears above a vowel, so its appearance is dependent on the next keystroke.

- **Goal**: Find words containing, but not starting with, *sp*
  Regular expression: `[^_]sp`
  Result:

  D ∧ *respekt, eksploderede, flammespil,* etc.

  E ∧ *aspects, hospital, transport, respond,* etc.

  G ∧ *weitersprach, ersparen, Glasplatte,* etc.

  S ∧ *española, disputas, responde,* etc.

  Comment: [^_] means 'not a space', so the regular expression given above means 'match words which contain any character, except a space, followed by *sp*'. (*Note*: Recall that _ is not a character which you type in; it represents a space, so press the space bar. In fact, you will have to press the space bar twice in a row to produce [^].)

- **Goal**: Find English words which contain letters other than vowels between *b* and *l*

Regular expression: `b[^_aeiouy]`l
Result: E ∧ *subtle, pebble, dabble,* etc.

Question 1: Why include space (_)in the set? How would the results differ if we did not include it? Try it and see.

Question 2: Note that <y> is included in the set, since it can sometimes be used as a vowel. In this case it is necessary because there are a few English words containing the string <byl>, and these would appear if this letter were excluded from the set. Can you think of any such words? Try a search on <byl> if you want help from the computer.

7.2 Parentheses () combined with other special characters

Parentheses are used to group strings so they are treated as a unit. When used with other special characters, they can provide interesting search patterns.

- **Goal**: Find words containing *ba* followed by at least one instance of *na*
  Regular expression: `ba(na)+`
  Result:

  E ∧ *banality, banana, Copacabana,* etc.
  S ∧ *urbana, Habana, cubana,* etc.

- **Goal**: Find instances of the forms of the irregular verb *go* in English
  Regular expression: `_go(_|es_|ne_|ing_)|_went_`
  Result: E ∧ *go, goes, going, gone, went*

  Comment: Note that this regular expression yields the same results as one listed earlier: _go_|_goes_|_gone_|_going_|_went_

- **Goal**: Find sequences of two or more 4-letter words which alliterate on initial *s* followed by a vowel
  Regular expression: `_(s[aeiou]\w\w_)(s[aeiou]\w\w_)+`
  Result:

  D ∧ *sund sans, samt seks, sagt selv,* etc.
  E ∧ *some such, soon sees, suit some sour,* etc.
  G ∧ sich seit, sein soll, sind sehr, etc.

7.3 Restrictions on combining special characters

It should be noted that most special characters lose their status as special characters when they appear inside square brackets. Thus an expression such as [ba(na)+\w] is simply a search for strings containing one (or more) of the following characters: *abnw\()+*.

On the other hand, the use of the caret (^) is pretty much restricted to use inside square brackets.

8. Test your understanding of regular expressions

Here are five examples of search goals, each of which can be met by writing a single regular expression. You might want to try to write regular expressions which will meet these goals. Since there are five goals, you should come up with five regular expressions, one for each goal. Suggested solutions are provided in Appendix 2.

- **Goal 1**: Find examples of German nouns in neuter gender using the Mannheimer corpus (mak)
- **Goal 2**: Find 4-letter words which rhyme with *shirt* in the English corpus (bnc)
- **Goal 3**: Find, in the Danish corpus (dfk), all occurrences of *vor tid, vores tid, vor fremtid,* and *vores fremtid* which are not preceded or followed by punctuation
- **Goal 4**: In the Spanish corpus (camtie) find sequences of two or more words in a row, each starting with a vowel, each consisting of at least three letters, and each ending in *s*
- **Goal 5**: Find the total number of occurrences of the words *computer, Computer, and COMPUTER* in the BNC. (Before you tackle this problem, read the two notes below.)

  Note 1: Your count should include the occurrences of *computer*, *Computer*, and *COMPUTER*, whether or not they are preceded or followed by punctuation marks, and only if they are not part of a longer string such as *computerised* or *computers.*

  Note 2: We are interested only in the total number of pattern matches, not in a list of occurrences (there are over 10,000 instances of *computer/Computer/COMPUTER* in the BNC, so listing them will take some time). Consequently, you should select 'just count' in the menu specifying the number of desired matches. (Of course, you may wish to see the first 10 or first 100 occurrences to determine the accuracy of your search string.)

Appendix 1

The following chart provides a summary of the regular expressions illustrated in sections 5 throgh 7. For each expression, sample results from the BNC are given.

Regular expression	**Sample results (BNC)**
ess	Professor, unless, expressed, ...
ess_	unless, access, stress, success, ...
_scr	scrutiny, scrap, script, ...
low	low
..ing	being, dying, going, thing, ...
_b..b_	bomb, bulb, blob, ...
dogs?	dog, dogs
"?nice"?	nice, "nice", "nice, nice"
st*le_	castle, apostle, isle, Carlisle, ...
et+ing_ ee|oo	getting, meeting, letting, marketing, ... blood, between, schools, need, ...
_go_|_goes_|_gone_|_going_|_went_	go, goes, gone, going, went
[aeiouy]_	immune, a, the, to, so, very, ...
_s[ptc] _[A-Z][a-z]	still, school, speak, ... This, How, ...
_[Aa]a	Aaron, aah, Aachen, ...
_[0-9][0-9]_	10, 40, 24, ...
etc\\.	etc.
st\wd	listed, studied, tested, custody, ...
_\w\w\w\wy-\w\w\w\wy_	fuddy-duddy, willy-nilly, fifty-fifty, ...
[^_]sp	aspects, hospital, transport, ...
b[^_aeiouy]l	subtle, pebble, dabble, ...
ba(na)+	banality, banana, Copacabana, ...
_go(_|es_|ne_|ing_)|_went_	go, goes, going, gone, went
(s[aeiou]\w\w)(s[aeiou]\w\w_)+	some such, soon sees, suit some sour, ...

Appendix 2

Here are some suggestions for regular expressions which will meet the goals set in section 8:

- **Goal 1**: Find examples of German nouns in neuter gender using the Mannheimer corpus (mak)
  Regular expression: `_das_`
  Result: G ∧ *das (Schicksal), das (Herz), das (Taxi),* etc.
- **Goal 2**: Find 4-letter words which rhyme with *shirt* in the English corpus (bnc)
  Regular expression: `_.[eiu]rt_`
  Result: E ∧ *hurt, Bert, dirt,* etc.
- **Goal 3**: **Goal 3**: Find, in the Danish corpus (dfk), all occurrences of *vor tid, vores tid, vor fremtid,* and *vores fremtid* which are not preceded or followed by punctuation
  Regular expression: `_vor(_|es_)(frem)?tid_`
  Result: D ∧ *vor tid, vores tid, vor fremtid, vores fremtid* (16 instances in all)
- **Goal 4**: In the Spanish corpus (camtie) find sequences of two or more words in a row, each starting with a vowel, each consisting of at least three letters, and each ending in *s*
  Regular expression: `(_[aeiou][^_]+s)(_[aeiou][^_]+s)+_`
  Result: S ∧ *esos aromas, estamos obligados, algunas investigaciones importantes*, etc.
- **Goal 5**: Find the total number of occurrences of the words *computer, Computer, and COMPUTER* in the BNC
  Regular expression:
  `[^a-zA-Z]([Cc]omputer|COMPUTER)[^a-zA-Z]`
  Result: The total number of occurrences = 10,051. (If you are interested in the breakdown, *computer* occurs 7643 times, *Computer* occurs 2260 times, and *COMPUTER* occurs 148 times.)