Lexicon Creator: A Tool for Building Lexicons for Proofing Tools and Search Technologies

Thierry Fontenelle
Nick Cipollone
Mike Daniels
Ian Johnson
Microsoft Corporation

In this paper, we describe Lexicon Creator, a tool designed to help developers produce lexical data for its use in a variety of linguistic applications such as spell-checkers, word-breakers, thesauri, etc. The tool enables developers to work on existing wordlists derived either directly from corpora or from previously created wordlist data. The key feature of the tool is that it enables linguists to rapidly create the morphological rules that are necessary to generate all the inflected forms of a given item. In many languages, a given word may have many forms, each distinguished by different endings attached to the stem of the word. A language like English is rather simple, morphologically-the verb walk only has the following forms: walk, walks, walked, walking, while other languages may have a number of different forms for a word. Yet, it is essential to create lexicons that can recognize and generate all the inflected forms of a given word, especially for applications such as spell-checkers-where overgeneration should be avoided, thesauri, grammar checkers, morphological analyzers/generators, speech recognition, and handwriting recognizers. It would be extremely time-consuming to code each of these forms individually, so it is necessary to develop this data more efficiently. Lexicon Creator allows linguists to classify these variations of the same word into templates, or morphological classes, which allow the automatic generation of all valid forms of a word. Once the templates describing the aforementioned variations have been defined, the data-coding task consists of assigning an input word to the correct template and checking that the forms generated automatically are valid. The article will also focus on the additional types of linguistic information which can be attached to words, depending on the intended application that will use the resulting full-form lexicon.

Introduction

Applications such as spell-checkers, morphological analyzers or handwriting recognizers rely upon large lexicons with varying degrees of linguistic information attached to the lexical items. In the case of spell-checkers or handwriting recognizers, one straightforward way of approaching the lexicon issue is to say that a list of all valid word forms is sufficient, which means that the lexicon should be as large as possible and should include all the word forms one would expect the tool to verify (i.e. accept and possibly suggest, in the case of a spell-checker). While this may be true for many languages in theory, it says nothing about how this list of possible word forms should be created. Knowledge of the morphological rules of a language is of course essential and care should be exercised when implementing these rules to avoid the overrecognition issues which are commonly found in NLP applications geared towards the analysis of unknown words. Fontenelle (2004) shows how careful the creators of spellers should be to avoid producing analyses for words which are spelling mistakes (e.g. *conu* being potentially analyzed as co+nu "co-naked" when the intended French word is *connu*, or *transfer* being potentially analyzed as trans+fer "transiron" when the correct spelling in French should be *transfert*). One solution we adopted for building the lexicons that underlie spell-checking technology and stemmers used in some search engines is to store lemmas together with their inflected forms in our lexicons. This means that we need to be able to generate all and only the

possible correct forms, which requires in-depth descriptions of the morphological rules of a language.

Building full-form lexicons

Compared to many other languages, English has a relatively impoverished morphological system. Nouns typically form their plurals by adding *-s* or *-es* (after a sibilant) and have a nearly non-existent case system, the genitive being the only remnant of such a system (*the president's decision*). This means that a given noun normally has 4 distinct forms (e.g. *dog, dogs, dog's, dogs'*). Verbs traditionally have 4 or 5 distinct forms (*work, works, worked, working; eat, eats, eating, ate, eaten*). It would be extremely time-consuming to list all these forms and add them to the lexicon individually, which is why we have developed a tool, *Lexicon Creator*, that enables linguists to author the morphological rules of a given language and facilitates the generation of inflected forms. *Lexicon Creator* allows linguists to classify variations of a given word into templates, which correspond to morphological classes. Once the templates have been defined, the data-coding process basically consists of associating a word to its correct template and checking that all the possible inflected forms of this word are generated (and only these inflected forms, if one wants to avoid overgeneration). If a template includes the rules that are necessary to create the distinct forms of a verb like *work* (infinitive = base form; 3rd person singular = *+s*; past tense = *+ed*; present participle/gerund = *+ing*), it will then be sufficient to associate other verbs such as *walk, talk, bark, crack…* with the same template to automatically add inflected forms such as *walks, talked, barking* or *cracked* to the lexicon, without having to manually list them.

Authoring morphological rules

Lexicon Creator provides a powerful authoring environment which enables the linguist to create sometimes very complex morphological rules, using regular expression patterns, character/string classes and functions which have made it possible to describe the inflectional system of over 40 languages, including Romance, Germanic, Scandinavian, Uralic, Semitic, African, Indic and Asian languages. Phenomena like gemination (En. *big – bigger*), vowel changes (Ge. *Buch – Bücher*), stem changes (Fr. *venir – viens*) can be expressed easily via functions which encode transformations consisting of patterns and replacements. Gemination of final consonants can be expressed as follows, for instance:

> Patterns appear to the left of the arrow and replacements appear to the right of the arrow.
> **GeminateCons** (e.g. *travel* → *travell*):
> (.*<:vowel:>)(<:gemcons:>) → (1)(2)(2)

This function matches zero or more characters (.*) followed by a vowel and a consonant and returns the same initial sequence of characters followed by the same vowel followed by two occurrences of the consonant. <:vowel:> has been defined as a string class consisting of all vowels. <:gemcons:> has been defined as a string class consisting of all geminating consonants (b, d, g, k, l, m, n, p, r, t).

Once such a function has been defined, it can be used in writing the rules that can generate the inflected forms that undergo gemination for a class of verbs that includes *stop, blog, dot, flip…* A rule which forms the past tense would then look like this:

> (GeminateCons<1>)ed

which can be read as follows: the GeminateCons function applies to Stem #1 and the suffix "ed" is added to the output of that function. Assuming that the stem is *stop*, the intermediate geminated form will be *stopp* and the final inflected form generated by this rule will be *stopped*. The same function would be useful to create rules for adjectives that double their final consonant (*big – bigger, dim – dimmer…*). It will be used in English but also in languages like Dutch, where gemination is also fairly common (*dun – dunner; kat – katten*). It should also be noted that there need to be (at least) two distinct templates, one for words that cannot undergo

gemination, and one for items that do (compare *stop* and *develop*, the former undergoing gemination in *stopped*, unlike the latter – *developed*).

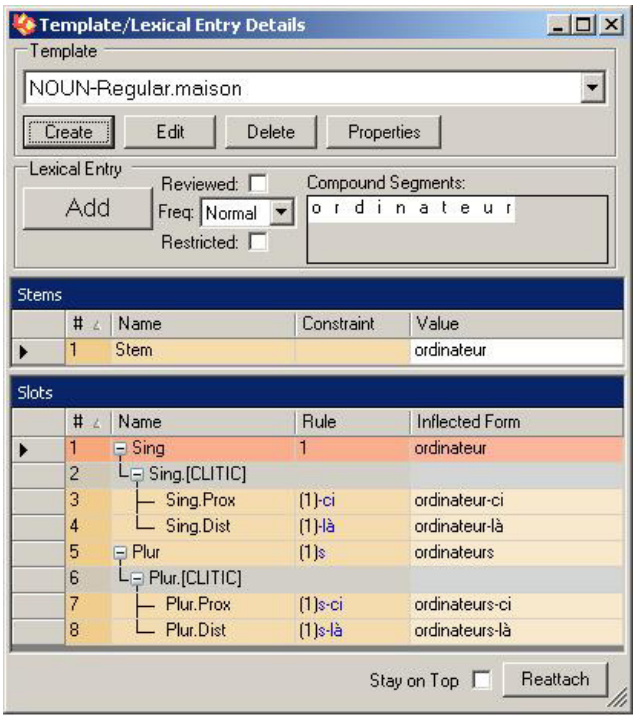
Adding entries to the lexicon

In our context, lexical acquisition means adding words to the lexicon, which boils down to assigning a template to a lexical entry so that all the correct inflected forms can be generated and stored in the lexicon. The lexicographer using *Lexicon Creator* can use a word list as a source of lexical data. Powerful sorting and filtering functionalities make it easy to target specific subsets of the word list or to sort it as a function of various criteria, for instance in increasing or decreasing order of frequency (if the list includes word forms extracted from a large corpus with frequencies of occurrences). Because the ending of a word can traditionally be used as a clue to the morphological properties of a lexical item, it is also possible to sort the word forms in reverse alphabetical order, which groups words that have a common ending. For instance, *whisky* and *baby* should be assigned to the same template since they both have a plural in *–ies* (*whiskies*, *babies*); similarly, *easy* and *happy* are similar insofar as they both have the same types of comparatives and superlatives (*happier/happiest, easier/easiest*). Words that share common morphological properties can be grouped together and bulk-imported via a dynamic user interface which guides the user through a series of menus to choose the appropriate template classes, templates and inflected form slot names to which these words should be assigned.

An "Analyses" window presents the list of possible ways of analyzing a given word form which are ordered according to how closely the word matches a particular pattern (the highest ranking match appearing at the top of the list). This window helps pick the correct template. The choice can be complicated by the multiplicity of templates in a language, especially when that language is morphologically richer than English. The languages for which we have created template lexicons with *Lexicon Creator* typically have several dozen morphological classes for nouns, adjectives and verbs and it is not always easy to choose the correct template which will generate the correct inflected forms of a given lemma. To facilitate the lexical acquisition process, the lexicographer can select multiple entries in the Word List panel, sort them and filter them according to various linguistic criteria and bulk-assign them to a given template when she is 100% sure (see screenshot below). In addition to these powerful functionalities, the tool also includes interactive lexicon merging and comparison features, which make it possible to compare two versions of the same lexicon and to merge changes made by different users.

Affix management via subtemplates

Templates can be viewed as either highly structured entities or as flat tables. In order to capture morphological redundancies, templates can have an internal substructure which allows for the management of regular affixes attached to inflected forms. Such substructures are intended to be used in languages where multiple, individually-separable affixes can attach to a single word in a "chain". By representing each affix as a separate data structure, linguists can build up structured templates that encode all possible affix combinations without having to enter and maintain multiple copies of the same information. Subtemplates can be embedded in the rules tree of a given template to specify how affixes are realized in different environments. In French, clitics *-ci* and *-là* are deictic locative adverbs corresponding to "here" and "over there" (*cet ordinateur-ci* "this computer here" vs. *cet ordinateur-là* "that computer over there") that can be attached (with a hyphen) to singular and plural forms of any noun. We can therefore define a general class of affixes for these deictic adverbs (subtemplates can also be used for non-hyphenated clitics, as in Spanish with *dámelo* "give it to me" or *dame* "give me", for instance).



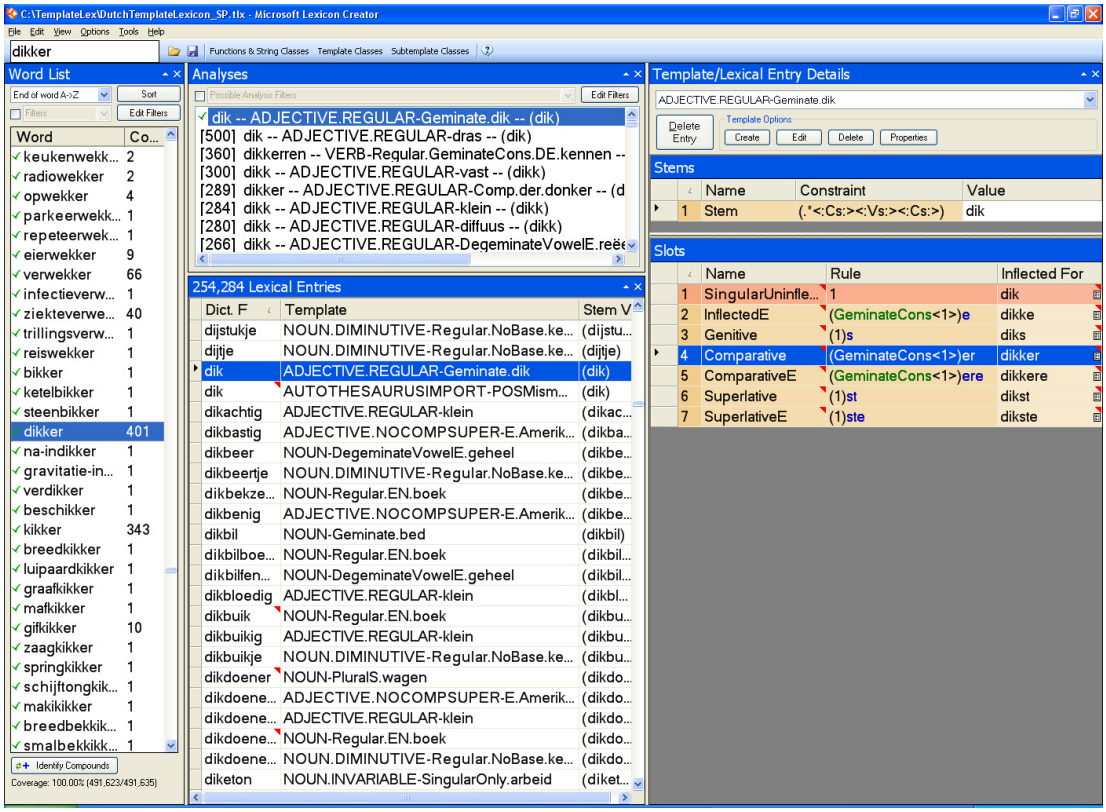
Lexicon Creator's panels

Lexicon Creator has four main panels, as shown in the screenshot below:

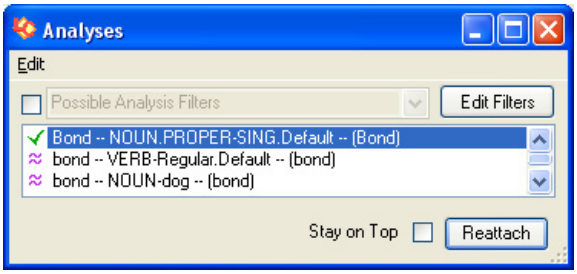
- Word List: this panel allows the lexicographer to load an external list of words (with optional frequencies) and view, sort and filter this list in a number of ways. The words from a corpus used as a source to populate our lexicons will typically be displayed and manipulated in this panel. Powerful filtering functionalities enable the lexicographer to quickly identify words which are or are not already in the lexicon, words which end in a given suffix, and even words which can be analyzed as valid compounds by the dynamic compounding module described below.
- Analyses: When a word is selected from the word list, it is automatically copied to the "current word" textbox in the toolbar. The system then generates a list of possible templates to which this word can be assigned. This list of possible analyses is presented in the Analyses window and is ordered according to how closely the word matches a particular pattern. If the word is in the lexicon, its actual analyses (the lexical entry or entries the word is associated with) are displayed at the top. On the basis of the information presented in this panel, the lexicographer can select the template that seems most likely to provide a correct description of all forms of the input word. When she clicks on the template she thinks is correct for this analysis, *Lexicon Creator* shows the word selected in the Template/Lexical Entry Details panel with all inflected forms generated by applying the rules defined when the template was created. She can then verify that the list of stem forms and inflected forms is correct.

  The tool is able to quickly locate inflected forms without having to fully generate the list via an inflected form indexer which allows us to interact with even enormous (multi-billion-word) lexicons without using much memory. In the example below, the Dutch adjective *dikker* is in the lexicon and the actual analysis indicates that it is associated with the lemma *dik* "big" assigned to the template ADJECTIVE-REGULAR.Geminate.dik.
- Lexical Entries: This panel displays the entries of the template lexicon. By default, it shows three columns: the lexical entries, together with the template which is assigned to them and the stem values.

- Template/Lexical Entry Details: This panel allows the lexicographer to review the forms which are generated when a given lexical entry is associated to a particular template, accessible from a drop-down menu. If the template generates all the correct forms for the word, segmentation information for compounds can be added if applicable and necessary, and the word can be added to the lexicon. In the example below, the form *dikker* in Slot #4 is the comparative of the adjective *dik*, which has other possible inflected forms: *dikke, diks, dikkere, dikst, dikste*. The rule applied to generate comparatives for that template makes use of the GeminateCons function which ensures that the final consonant is doubled before the comparative suffix *–er* is added (*dik-k-er*).



Words that are already in the lexicon appear in the Word List with a green checkmark. Working with corpus data to derive word lists forces the lexicographer to consider the normalization strategies she would like to use. Since word lists extracted from corpora can include capitalized and non-capitalized versions of the same word (*drink* vs. *Drink*; *john* vs. *John* vs. *JOHN*...), the lexicographer may want to know which forms are covered by her lexicon by using the concept of Extended Actual Analyses. A purple approximately equal (≈) sign appears next to words whose "extended actual analysis" is in the lexicon, i.e. words whose normalization is in the lexicon (*DRINK* or *Drink* would get this symbol if *drink* is in the lexicon). The same system is used to display the possible analyses of an input string, and it is possible for a given string to be analyzed in multiple ways, resulting in multiple word-template pairs, as in the following example illustrating the analyses of *Bond*:



The string *Bond* can correspond to the proper noun (James) Bond; the approximately equal sign indicates that *Bond* has an extended actual analysis (normalization) which is in the lexicon (both as a verb entry and as a regular noun entry).

Annotating lexical entries

The main purpose of *Lexicon Creator* is to be able to associate a lemma to all its valid inflected forms, using the concept of template or morphological classes. It is possible to enrich the lexical data, however, by annotating lexical entries or their inflected forms with linguistic attributes. In addition to part of speech information, the data can be annotated with information that can be consumed by applications such as spell-checkers, thesauri, morphological analyzers, handwriting recognizers, *viz*:

- Dialect/regional distinctions (e.g. US vs. UK vs. Canadian vs. Australian English…; old vs. new spelling for languages that have undergone a spelling reform, like French, German or Dutch; such information is crucial to ensure that the reform settings selected by the user of Microsoft Office spellers, if any, produce the desired results, as is shown in Fontenelle 2006)
- frequency information
- levels of formality (offensive words may undergo a special treatment in a speller)
- linking spelling variants (for query expansion in search engines, which need to know that a form like *produkt* in Dutch is the old spelling of what should now be written *product* after the 2005 spelling reform; similarly, the French words *ile* and *île* can be linked as spelling variants for search technologies)
- compound segmentation (for compounding languages, knowing where a lexicalized compound should be segmented is crucial in order for a word-breaker to be able to emit the correct segments in a search and indexing perspective)
- synonyms (for thesauri)
- number, person, tense, gender, transitivity information, etc.

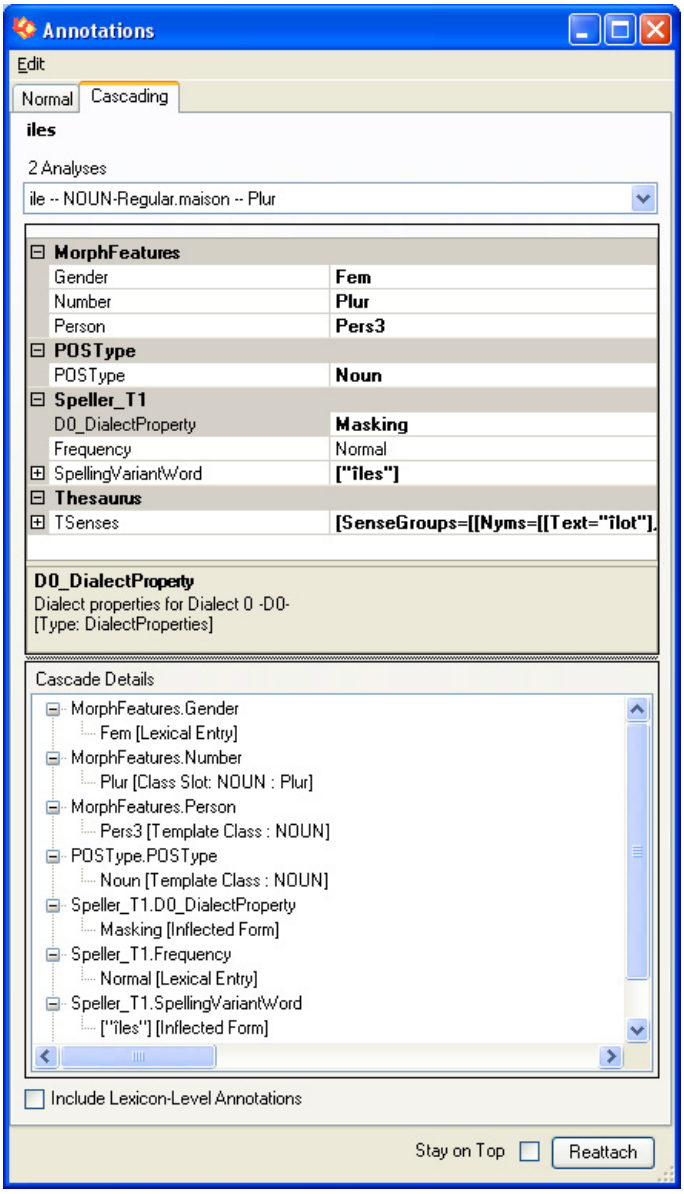
Any type of linguistic information that can be useful for NLP, including text mining or classification schemes, can be added via flexible annotation schemas.

Depending upon the type of application the linguist is developing, she will have to create an annotation schema which defines the additional types of information (annotations) she would like to be able to use. Once an annotation schema has been defined with the Annotation Schema Editor, she can then load the schema, associate it with a lexicon and start annotating the data, which means that sets of attribute-value pairs can be attached to lexicon objects. A schema associated with a lexicon is embedded as a resource inside a template lexicon. Schemas can be modified and upgraded (for instance when a given application requires a new type of linguistic information, which needs to be added to an existing schema).

Any kind of lexicon object can be annotated: template classes or templates, lemmas (lexical entries), individual inflected forms, or even slots corresponding to inflected forms associated with a given part of speech (e.g. one might decide to annotate all inflected forms corresponding to a Past Subjunctive in French as Frequency=Low, for instance). Because annotations can be assigned at various levels, it has been necessary to develop a system of "cascading annotations". This system interprets annotations by assigning a total order on objects in the lexicon and defining rules for annotation overriding and inheritance. Annotations "cascade" from a higher lexicon object to a lower one and lower values on the cascade override higher values. For instance, a noun inflection might have Restricted=Archaic set at the template class slot level, with that inflection set to Restricted=None at the (lower) inflected form level of a few words where the inflection is still commonly used.

In the example below, which illustrates the cascading annotations of the French plural word form *iles*, one can see that the annotation Number=Plur is inherited from a Slot called Plur (the value

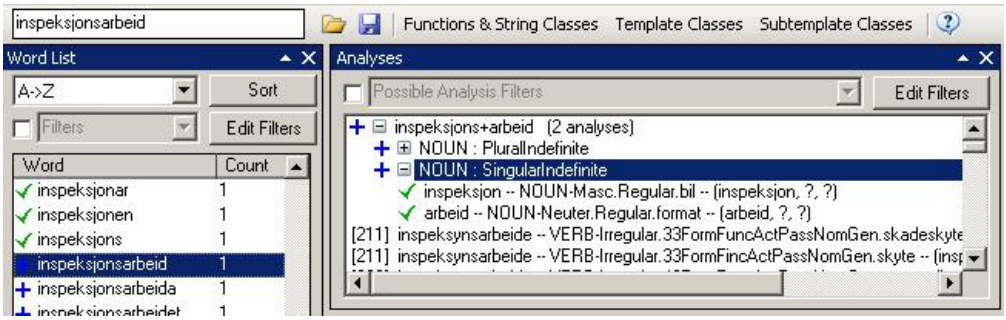
Plural cannot be inherited from the lemma (Lexical Entry) *ile*, unlike the Gender attribute, which is assigned at the lexical entry level and percolates down to all the inflected forms associated with this lemma). Similarly, the Part of speech (POS) type "Noun" is inherited from the attribute-value pair POSType=Noun assigned to the Template Class NOUN.



Dynamic compounding

The tool also allows the lexicographer to define legal compound structures for compounding languages. Information about whether a given word form can act as an initial, medial or final segment in a compound can also be stored in the lexicon to make sure that the speller handles dynamic compounding. This is crucial for compounding languages such as German, Dutch, Swedish, Norwegian or Danish, since it is impossible to lexicalize all possible compounds. A dynamic recognition mechanism is therefore necessary to make sure creative (non-lexicalized) but valid compounds do not get flagged by a spell-checker. The linguist can experiment with compound analysis and visualize the results in the Analyses panel to simulate the behavior of the speller engine. The following screenshot illustrates the Norwegian Nynorsk word *inspeksjonsarbeid*, which is not in the lexicon and gets 2 possible compound analyses according to the segmentation rules of the language. A big blue + sign signals the existence of compound

analysis. Analyses with fewer segments (which are *a priori* better analyses) are displayed first. The analysis shows that this non-lexicalized compound is analyzed as the noun *inspeksjon* followed by the noun *arbeid*.



Dynamic compounding logic can be defined directly in the template lexicon. Declarative descriptions of legal compound structures are entered via an editor: these "segment rules" are in fact regular expressions over literal characters and classes of entries in the lexicon. The classes are defined by "segmentation bits", which are present on every runtime lexicon entry. Encoding dynamic compounding for a language thus reduces to:

1. deciding what classes of lexical entries there need to be;
2. writing the correct segment rules to combine these classes of lexical entries;
3. populating the lexical entry classes appropriately – in other words, assigning the segmentation bits.

One such rule in Norwegian could stipulate that valid compound structures have the form <Seg1><Seg2>*<Seg3>, which means that valid compounds are made up of an initial segment Seg1 followed by 0, 1 or more medial segments marked as Seg2 followed by a final segment marked as Seg3. In the example given above, this means that the inflected form *inspeksjons* will need to be annotated as Seg1 (like all word forms that can be initial segments) and *arbeid* will need to be annotated as Seg3 in the lexicon (like all word forms that can be a final segment). These segmentation annotations would of course be annotated on the most general applicable level, often the template class slot level, to then cascade down to the individual inflected forms. The runtime engine driving the spell-checker will include the same logic to determine the valid compound patterns and flag those compounds that are not permitted by this logic. In addition to the definition of segment classes, *Lexicon Creator* also makes it possible to configure rules according to the number of segments that can enter a valid compound, as well as to the length of segments (e.g. for a given language, the linguist might specify that each segment should be at least 3 characters long) or the minimum length of a non-lexicalized compound.

Derivational morphology

Lexicon Creator provides functionality to capture phenomena related to derivational morphology, which creates new words (usually, though not necessarily, of a different part of speech) by adding a bound morpheme to a base form. For example:

Adj → Adv in -ly: *quick* → *quickly*

Adj → Verb in -en: *bright* → *brighten*

A flexible environment enables the linguist to create derivational rules and to specify constraints allowing or blocking the application of these rules to limit overgeneration. One could for instance indicate that a given rule applies only if the source lexical entry is not tagged as Vulgar.

Each derivational rule consists of three parts:

1. constraints on the source lexical entries (the left-hand side of the rule)
2. output lexical entry specifications (the right-hand side of the rule)
3. the rule's application mode

Constraints can be expressed via powerful filtering mechanisms using regular expressions and annotations. A rule that would generate adverbs in -ment in French could for instance constrain its application to the feminine singular form of adjectives (*public* → Fem. *publique* → Adv. *publiquement*; *fier* → Fem. *fière* → Adv. *fièrement*). The resulting entries (output of the derivational rule) can be specified through a set of stem mapping rules based on stems or on slots corresponding to an inflected form (e.g. Slot3(ment) if Slot3 corresponds to the feminine singular form of an adjective). Annotations can also be automatically attached to the output of a rule (e.g. specifying a given gender for all the derived forms produced by a given rule).

Three rule modes can be chosen to specify how the rule should be applied:

- OptIn: the rule does not apply by default but can be turned on
- OptOut: the rule applies by default, but can be turned off
- Obligatory: the rule applies by default and cannot be turned off

For the first two modes, the linguist must specify where the rule applies (or where it does not apply) while obligatory rules apply without exception to all lexical entries admitted by their filters. In the case of the French rule referred to above, for instance, it would be necessary to block the derived form *brèvement* (*bref* → Fem. *brève*), since the adjective *brièvement* (briefly) is used instead.

Inflected form filters and lexicon scanning

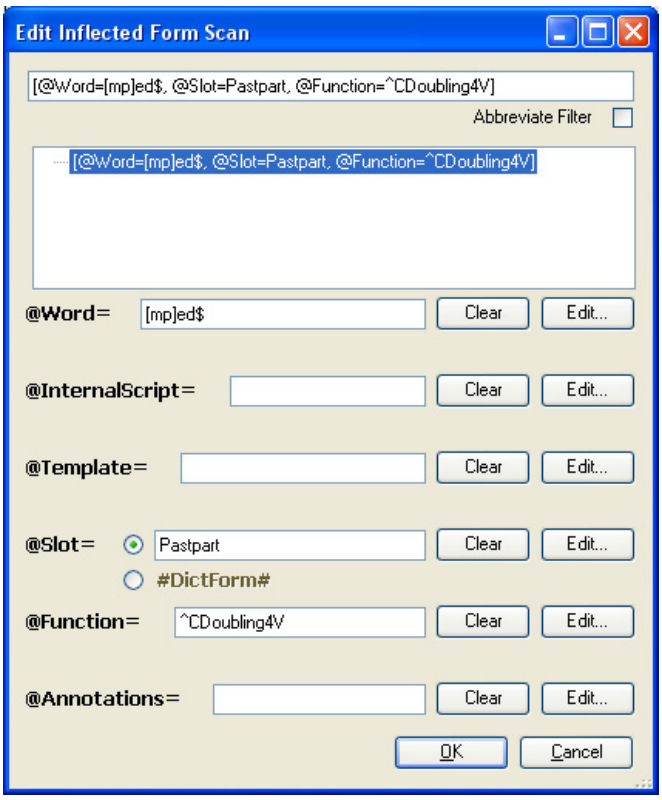
Building a generic lexicon for multiple applications comes with a set of challenges from a lexicon management perspective. One of the crucial questions is how to store multiple, partially overlapping versions of lexicon codes and attribute-value pairs in a single lexicon so that different clients of the lexicon (i.e. applications that use it) can retrieve different versions without having one client's requirements pollute the shared, generic lexicon. To solve that issue, we have implemented a declarative interface layer in the lexicon that translates mappings between client notions and lexicon structure. Sets of named inflected form filters can be used to define sets of part of speech tags, syntactic or semantic bits, etc., that will persist in the template lexicon for various applications. A specialized UI has been included for such cases. For a specific application, for instance, it might be useful to a create a particular POS tag set which maps masculine singular past participles (found in a verbal template) onto an Adjective_Masculine_Singular tag, without having to specifically annotate these verb forms as adjectives. The specific tag set would be used by a thesaurus, for instance, while another application would preserve the distinction between verbs and adjectives. The named filter set would then only be exposed to the applications that need it.

The lexicographer can scan inflected forms. It is possible to look for words whose inflected forms match a given criterion, using standard regular expressions. A lexicographer working on an English full-form lexicon might want to use this function to carry out a consistency check and see whether the lexicon includes inflected forms ending in *-shs* or *-chs*, for instance. This could point to mistakes that need to be corrected (e.g. **watchs*).

Another potential application of such a scan is to find whether prefixes or suffixes have been used in template rules with a different script than the one used for dictionary forms. For instance, in a lexicon using the Cyrillic script, one might want to identify possible mistakes in words for which Latin characters might have been used erroneously in suffixes or prefixes.

This scanning functionality can be used to carry out consistency checks, study the distribution of some linguistic phenomena and browse the lexicon in an opportunistic mode. Filters on

inflected forms can be defined via a dialog box which automatically builds regular expressions for the user, as in the screenshot below, which illustrates a query to list inflected forms which correspond to a Pastpart slot (=past participle) ending in -med or -ped ([mp]ed$) and which use a function to double the final consonant of the lemma (in this example, this function is called CDoubling4V). Such a query will generate word forms such as *snapped, capped, chipped, chopped, brimmed, dimmed, unzipped, worshipped*…



Conclusion

Dictionary writing systems are increasingly used to compile dictionaries from corpus data (see, for instance, Joffe and de Schryver 2004). As noted by Kilgarriff (2005b), "a dictionary is a highly structured document and an entry typically contains a headword, pronunciation and part of speech code, optional labels and information about inflectional class and morphological and spelling variants, then a sequence of senses, each with definition or translation and optional examples. Each of these is a different information field." *Lexicon Creator*, as described in this paper, shares a number of features with some off-the-shelf dictionary writing systems and it can handle linguistic phenomena from a wide variety of languages, including Romance, Germanic, Slavonic, Scandinavian, Indic, Asian, African, Uralic, and Semitic languages. Its unique feature is its ability to allow lexicographers and non-experts in computational linguistics to author the morphological rules of a language and to create (and store) lexical entries together with all their inflected forms, a crucial functionality for applications such as spell-checkers, thesauri, and stemmers for search engines, speech and handwriting recognition systems, which require specific linguistic information about inflected forms, and not just about lexical entries.

References

Fontenelle, T. (2004). “Lexicalization for proofing tools”. In Williams, G.; Vessier, S. (eds.) *Proceedings of the 11th Euralex International Congress*. Lorient: Université de Bretagne-Sud. 79-86.

Fontenelle, T. (2006). “Developing a Lexicon for a new French Spell-checker”. In Corino, E.; Marello, C.; Onesti, C. (eds.) *Proceedings of the XIIth EURALEX International Congress.* Turin: Università di Torino. 151-158.

Joffe, D.; Schryver, G. M. de. (2004). “TshwaneLex, a state-of-the-art dictionary compilation program”. In Williams, G.; Vessier, S. (eds.) *Proceedings of the 11th Euralex International Congress*. Lorient: Université de Bretagne-Sud. 99-104.

Kilgarriff, A. (2005a). “Informatique et Dictionnairique”. Numéro spécial de la *Revue Française de Linguistique Appliquée* (RFLA) sur les dictionnaires 10 (2). 95-102.

Kilgarriff, A. (2005b). “Use of Computers in Lexicography”. In *Encyclopedia of Language and Linguistics*. Oxford: Elsevier.