Aalborg Universitet

AALBORG UNIVERSITY
DENMARK

Implementation of Danish in the Natural Language Generator of Angus2

Larsen, Søren Støvelbæk; Fihl, Preben; Moeslund, Thomas B.

Publication date:
2007

Document Version
Publisher's PDF, also known as Version of record

Link to publication from Aalborg University

Citation for published version (APA):
Larsen, S. S., Fihl, P., & Moeslund, T. B. (2007). *Implementation of Danish in the Natural Language Generator of Angus2*. Computer Vision and Media Technology Laboratory (CVMT), Aalborg University.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal -

Take down policy
If you believe that this document breaches copyright please contact us at vbn@aub.aau.dk providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from vbn.aau.dk on: March 28, 2024

Implementation of Danish in the Natural Language Generator of Angus2

Søren S. Larsen, Preben Fihl, and Thomas B. Moeslund
Labratory for Computer Vision and Media Technology, Aalborg University
Niels Jernes Vej 14, 9220 Aalborg Øst, Denmark

I. Introduction

The purpose of this technical report is to cover the implementation of the Danish language and grammar in the Angus2 software. This includes a brief description of the Angus2 software, and the Danish grammar with relevance to the implementation in Angus2, and detailed description of how it is implemented.

II. Brief overview of Angus2

Angus2 is developed as part of a system capable of generating natural language output based on video sequences. Currently the video sequences in Angus2 is comprised of four scenes from surveillance cameras, two filming different crosswalks with pedestrians, one filming across a gas station, and one filming an intersection. Each of these video sequences features multiple objects and actors, that can be tracked and their actions written in natural language. The input for Angus2 is the output from a conceptual subsystem, where a situation is inferred by using a Situation Graph Tree (SGT) and a Fuzzy-Metric Temporal Horn Logic (FMTHL) inference. An illustration of the system can be seen in Fig. 1.

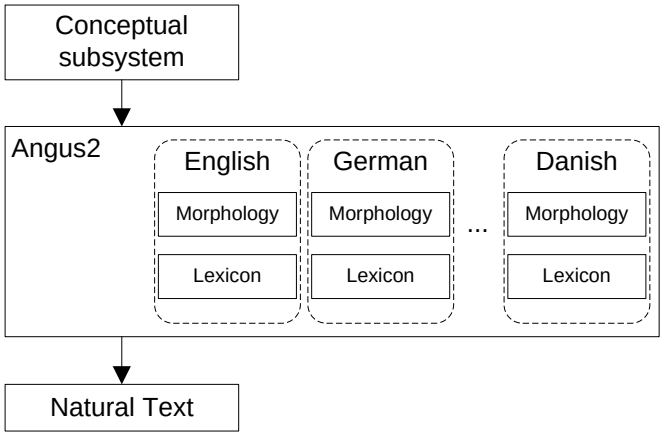


Fig. 1. Illustration of the Angus2 software, describing the subsystems and how they interact.

It is currently implemented in English, German, Czech, Catalan, Castellan, and Japanese. Screenshots of the Angus2 software can be seen in Fig. 2 and 3. It is based on Prolog and Java, but for implementing an additional language, only Java is necessary. [1] The IDE used is of little consequence,



Fig. 2. The initial GUI screen of angus2 after a image sequence is selected.

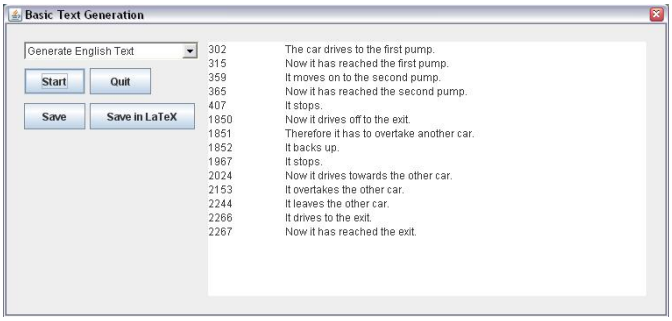


Fig. 3. The text generation menu of Angus2 after English text generation of a scenario.

and the free IDE Netbeans was used in implementing the Danish language in Angus2, however other free environments such as Eclipse is more than adequate for implementing an additional language. It is recommended however, that the source code is imported into a project in the IDE for easy management of the source code.

III. Danish Grammar

This is meant as a brief description of the Danish grammar. The most important aspect of the grammar with regards to

Angus2 is the sentence structure. The Danish language shares many common traits with both German and English. The differences is highlighted in this section.

A. Sentence Structure

The sentence structure of the Danish language is predominantly in a subject-verb-object (SVO) typology as in the English languages e.g. *"The house is blue"* (subject-verb-object). It is not true in all cases though, as Danish changes the order to verb-subject-object (VSO) when the sentence is started with an adverb (as in German) or when asking a question [2]. An example of this could be;

"From 10 to 12 I play soccer."
"Fra kl. 10 til 12 spiller jeg fodbold."

where red is the subject and green is the verb.

B. Conjugated Nouns

As opposed to many other languages, in Danish nouns and adjectives often comprise a single noun. The rule of thumb is that if the adjective describing the noun is a verb or a noun it is supposed to be a single word e.g. *"tram stop"* would be *"sporvognsstoppested"* (tramstop).

C. Definite Article

Another difference in the grammars of Danish and English is the way definite article is handled. In english the word "the" covers all the cases of definite article. In Danish there are two cases besides the difference in the two genders (common and neutrum). The lemma of a noun is suffixed with *"en"* or *"et"* depending on gender e.g. *"the car"* is *"bilen"*. When the noun is described by an adjective however, the sentence is prefixed with *"den"* or *"det"* e.g. *"the white car"* is *"den hvide bil"*.

IV. Implementing Danish Sentence Generation

The initial steps of implementing Danish sentence generation in Angus2 consist of simply replicating the English text from a Danish set of functions. This is feasible as the Danish and English grammars are quite similar. The sentence generation functions from the `sources.nlg.microplanning.drtMicroplanning.drs_transforming.sentence_building.english` and `sources.nlg.microplanning.drtMicroplanning.drs_transforming.sentence_building.english.transormation_rules` packages are simply copied and refactored into Danish equivalents, including filenames, imports and typecasts (`DRS2ENGConverter` to `DRS2DANConverter`).

In order to make the new language accessible in the application a few files, generic for all languages such as the graphical user interface, must be modified. To make an additional language a selectable option in the *"Basic Text Generation"* dialog, it is added as `languageChoice.addItem("Generate <Language> Text")` in the `BasicNLGDialog.java` file in the `sources.gui.nlgnlu.basic` package. Furthermore, for the option to perform operations, `DANISH` is added to the `Language` enumerator in the `NaturalLanguageTextGenerator.java` file in the `sources.nlg` package. It should be noted that the order of the languages in the enumerator `Language` should be the same as the order of the `languageChoice`.

In the file `DRS2Transformer.java`, the newly created package `sources.nlg.microplanning.drtMicroplanning.drs_transforming.sentence_building.danish.DRS2DANConverter` is imported and the function `DRS2DANConverter()` in the variable `allTransformers`. Again, the order of the functions is the same as in the prior mentioned variables.

Applying the steps above results in a compilable application with a functional Danish language option. However, as there is no available Danish vocabulary (nouns, verbs, adverbs etc.) and morphological rules, the application will produce errors. The rules and vocabulary are found in the files `morphology.mor` and `lexicon.lex` respectively. These files are in UTF-16 format, so when editing them, the editor must be capable of saving in this format as well. For Windows users the trial-ware text editor Editplus is recommended.

A. Morphology.mor

The file `morphology.mor` contains flexions, conjugations and other morphological operations applied to the lemma words (e.g. the lemma *"walk"* becomes *"walks"* in the present tense). Verbs are conjugated for present tense, perfect tense or special forms such as a requirement. The morphology.mor file is read by a simple parser, and the contents look like this:

```
VERB ENGLISH   PRES : VERB{be}is
VERB ENGLISH   PRES : VERB[o]es
VERB ENGLISH   PRES : VERB[s]es
VERB ENGLISH   PRES : VERB{y}ies
VERB ENGLISH   PRES : VERBs
VERB ENGLISH   PERF : AUX + VERBed
VERB ENGLISH   REQU : AUX + VERB
```

The variable `AUX` are auxiliary words required in the perfect tense and in the special requirement form. The words in `AUX` are found in the lexicon.lex file, and in English it is comprised of *"must"* and *"has"*. If multiple rules are required for correct grammar the order of these rules are important, as the parser runs through the rules as written. This means that if a word satisfies multiple rules, the first rule is applied. For example if the order of the above rules in present tense are reversed the verb *"be"* would become *"bes"* as opposed to the correct word *"is"* or *"go"* turning into *"gos"* instead of *"goes"*.

B. Lexicon.lex

The file `lexicon.lex` contains the lexication rules for the implemented languages. The SGT predicates have to be

related to nouns, adjectives, verbs, adverbs, prepositions etc. The entries in lexicon.lex looks like this:

```
ENGLISH IRREGULAR VERB come (Agent) {
  PREP in (Agent) {
    PREP from (DAT: Object) {
      enterfrom(Event,Agent,Object)
    }
  }
}
```

The structure of Danish versions of the entries in `morphology.mor` and `lexicon.lex` share great resemblance with the English entries, therefore the Danish entries are based on the English ones. Concretely the English entries are copied and the `ENGLISH` in the copies are substituted with `DANISH`, enabling Angus2 to find the proper morphology rules and nouns, verbs, etc.. Angus2 is now capable of reproducing English text when selecting the *"Generate Danish Text"* option in the *"Basic Text Generation"* dialog.

Translating the contents in morphology.mor and lexicon.lex from English to Danish directly, allows for a language generation that resembles Danish, but with obvious grammatical errors. The faults in the resulting Danish text are caused by the sentence inversions in the Danish language, the difference in perfect tense, the conjugated nouns, and the difference in the way definite article is handled.

In order to rectify these errors, it is necessary to alter some of the classes copied from the English sentence builder. The first thing is to change the java extension `extends` of `DRS2DANConverter` from `DRS2GerEngConverter` to the more generic `DRS2LanguageConverter`. The class `DRS2LanguageConverter`, contains general function prototypes for use in all languages. Inheriting this as opposed to `DRS2GerEngConverter` enables the developer to customise the functions to the Danish language. As the transformation rule classes still rely on many of the functions found in the `DRS2GerEngConverter`, the `add*`[1] functions from `DRS2GerEngConverter` are copied into `DRS2DANConverter` and used as templates for the modifications needed.

The order of the words in the sentences are defined by the order of the transformation rules entered in the `defineTransformationRules()` function. The order for creating natural english sentences is:

```
TR_FIND_SUBJECT_AND_OBJECT()
TR_FIND_SUBJECT()
TR_SUBJ_OBJ_WITH_SAME_ATTR2()
TR_SUBJ_OBJ_WITH_SAME_ATTR()
TR_DESTINATION()
TR_REQUISITE()
TR_STARTING()
TR_SUBJECT_PRONOUN_FEMALE()
TR_SUBJECT_PRONOUN_MALE()
TR_SUBJECT_PRONOUN()
TR_SUBJECT_DEFDESCRIPTION_FEMALE()
TR_SUBJECT_DEFDESCRIPTION_MALE()
TR_SUBJECT_DEFDESCRIPTION()
TR_SUBJECT_PROPERNAME()
TR_SUBJECT_NAME()
TR_VERBDESTINATION()
TR_VERBREQUISITE()
TR_VERB()
TR_VERBPREP()
TR_VERBPREP()
TR_OBJECT_NAME()
TR_OTHER_PERSON()
TR_OTHER()
TR_OTHER2()
TR_DELETE_EVENT()
TR_DELETE_EVENT2()
```

As seen, the entries are in SVO order, making the sentences in this order accordingly. As this is not always correct or sufficient for Danish grammar, it is needed to reorder and replace some of the transformation rules. German grammar allows for a different ordering of the sentences, so copying some of the transformation rules from german and applying them in Danish is a feasible option. The relevant classes copied are `TR_VERBZIEL`, `TR_VERBVORAUSSETZUNG`, `TR_VERBBEGINN`, `TR_VERBZIEL2`, `TR_VERBVORAUSSETZUNG2`, and `TR_VERBBEGINN2`. The first three java classes are added after `TR_STARTING`, while the last three replaces the existing `TR_VERBDESTINATION()` and `TR_VERBREQUISITE()` (Subsequently the German java classes have been renamed into English). This is used when writing sentences in perfect tense.

Writing in perfect tense requires the addition of an additional classification of morphology, as verbs in perfect tense are morphed differently in Danish. As a consequence, perfectum participium is added as a class of words in morphology.mor named PERP. This is used in the `TR_VERBDESTINATION2()` (previously `TR_VERBZIEL2()`) transformation rule.

As mentioned in Sec. III-A, the Danish grammar makes use of sentence inversion in cases where the sentence starts with an adverb. In order to handle this, the transformation rule `TR_INVERSION()` is added after `TR_STARTING()`, and the functions `setInversion()`, `isInversion()` and `resetInversion()` have been implemented in `DRS2DANConverter.java`. These functions controls a flag set when the transformation rule `TR_STARTING()` is called as it places an adverb at the start of a sentence. When this flag is set, the rule `TR_INVERSION()` is applied. This transformation rule is a modified version of `TR_VERB()` with the added functionality of checking the inversion flag with `isInversion()`. If this is set, it places the verb prior to the noun and resets the flag with `resetInversion()`.

[1] the asteriks '*' is used as a wildcard for additional functions.

Correcting the grammatical errors generated from the differences in the definite articles, as stated in Sec. III-C, is done by adding an `addARTtoNoun()` function that returns the proper determiner. The function is a modified version of `addDETtoText()`. It is called from the `TR_SUBJECT_NAME` class and is used when the noun is described with an adjective e.g. *"Den hvide bil"* (*"The white car"*). Furthermore, in the English version of `TR_SUBJECT_NAME` the noun added to the text when not described by an adjective, the function used is `addToText()` rather than `addNOUNToText()`, thus bypassing the morphology in morphology.mor as it is not needed in the English language. As this is needed in Danish, the function call is changed to `addNOUNtoText()`.

The problem with conjugated sentences is corrected in the lexicon.lex file, where the adjective and the corresponding noun is added as a single noun. This poses a problem however, as streetnames and names in general will be subjected to the same morphology as other nouns, which is not applicable in Danish grammar. An example of this is the English sentence *"The white car comes in from the Bernhard Street"* generated by Angus2. The generated Danish equivalent is *"Den hvide bil kommer ind fra Bernhardstrassen."*, whereas it should have been *"Den hvide bil kommer ind fra Bernhardstrasse."*, with the street name not subjected to the same morphology as other nouns. No solution for this problem has been implemented.

V. Discussion

The Danish language has been implemented in Angus2 over the course of two man weeks with approximately one week of getting to know the source code and design, and another implementing the language. The text generated from the cases in the data generated from the video sequences is correct in a natural Danish in most of the cases. The only apparant grammatical error still present is the morphology applied to names. Correcting this will likely require the addition of a noun subclass for special cases such as names, allowing a bypassing the morphology for example by using `addToText()` when a name is detected.

Enabling the possibility for more sequences of varying kind should prove to be a trivial task from a language generation point of view. This task involves adding the needed vocabulary through the `lexicon.lex` and possibly adding more morphological rules in `morphology.mor`.

References

[1] A. Fexa, J. Lang and I. Pop, *Angus2 v5.0 Reference Manual v1.1.2*, 2007.

[2] Wikipedia, *Subject Verb Object*, http://en.wikipedia.org/wiki/Subject_verb_object