Universidade do Algarve. Faculty of Human and Social Sciences
Universitat Autònoma de Barcelona. Facultat de Filosofia i Lletres
Université de Franche-Comté. Centre Tesnière

International Masters
in Natural Language Processing & Human Language Technology

# *Compansion* system for a pictogram-based AAC application in Catalan

PAHISA-SOLÉ, Joan
May 2012

**Dissertation supervisors:**
BAPTISTA, Jorge (UAlg)
BLANCO, Xavier (UAB)
CARDEY-GREENFIELD, Sylviane (UFC)

## ABSTRACT

In this project we have built a *compansion* (compression-expansion) system that expands telegraphic language (utterances with only uninflected content words) into natural language sentences in Catalan, in the context of an Augmentative and Alternative Communication (AAC) software.

The system has been designed to improve the communication of AAC users, who usually have severe speech or motor impairments and use pictogram-based communication methods in their daily life. However, it can also be seen as a potential tool to support Catalan literacy for these users and to help in the field of language rehabilitation.

The *compansion* system has two main components: a syntactic-semantic dependency parser and a generator that constructs the final sentence. A simile of an input system has been built to allow the use of an existing AAC vocabulary (CACE). This vocabulary has been enriched by syntactic and semantic information in view of the parser and the generator. From the user's input, the parser extracts sentence patterns, based on a constraint grammar and a controlled vocabulary, taking into account input words and their modifiers. It then assigns the remaining words to the structural slots of those patterns, which correspond to the semantic roles of the predicative elements and of other words' complements. Then, the generator builds the natural language sentence, from the resulting parse tree, reordering and inflecting words and adding articles, prepositions and other necessary particles.

The system has been evaluated and results are most promising: 98,7% of the sentences generated by the *compansion* system were considered correct by three independent annotators.

**Keywords:** Augmentative and alternative communication, natural language processing, *compansion* (compression-expansion), telegraphic language, Catalan, pictograms, dependency parser, semantic parser, controlled grammar.

## RESUMO

Este projecto consiste na construção de um sistema de *compressão-expansão* (em inglês: *compansion*) que expande frases em linguagem telegráfica, isto é, frases formadas apenas por palavras plenamente significativas e sem informação de flexão, em frases em língua natural (Catalão), no contexto de uma aplicação em Comunicação Suplementar e Alternativa (CSA).

O sistema foi projectado para melhorar a comunicação dos utilizadores de CSA, que normalmente apresentam graves perturbações motoras ou de fala e que usam métodos comunicativos baseados em pictogramas para as suas actividades quotidianas. Contudo, o sistema pode igualmente ser considerado como uma potencial ferramenta para apoiar a literacia em Catalão destes utilizadores e para ajudar na reabilitação da linguagem.

O sistema de *compansion* tem duas componentes principais: um analisador sintáctico-semântico (parser) e um gerador, que constrói a frase de saída. Um *simile* de sistema de input foi construído de forma a permitir a utilização de um vocabulário de CSA já existente (o CACE). Este vocabulário foi enriquecido com informação sintáctica e semântica para poder servir tanto ao módulo de análise como de geração. A partir do input, o parser extrai os padrões frásicos, baseados numa gramática e vocabulários controlados, e com base na informação associada às palavras e seus modificadores. Em seguida, atribui as palavras remanescentes às diversas posições estruturais desses padrões frásicos, que correspondem aos papeis semânticos associados aos diversos elementos predicativos e aos complementos de outras palavras. Em seguida, o sistema gera a frase de saída a partir da árvore de análise resultante, reordenando e flexionando as diversas palavras, acrescentando artigos, preposições e outros elementos necessários a uma expressão adequada em língua natural.

O sistema foi avaliado e os resultados são muito promissores: 98,7% das frases geradas pelo sistema foram consideradas correctas por três anotadores independentes.

**Palavras-chave**: comunicação suplementar e alternativa, processamento de linguagem natural, *compansion* (compressão-expansão), linguagem telegráfica, catalão, pictogramas, analisador de dependências, analisador semântico, gramática controlada.

## RESUMO ESTENDIDO

Este projeto teve como objetivo a construção de um sistema de compressão-expansão (em inglês: *compansion*) que expande frases em linguagem telegráfica, isto é, frases formadas apenas por palavras plenamente significativas e sem informação de flexão, em frases em língua natural (Catalão), no contexto de uma aplicação em Comunicação Suplementar e Alternativa (CSA). Os dois principais componentes do sistema são um analisador sintático-semântico (parser) da linguagem telegráfica e um gerador que constrói a frase de saída em língua natural.

No início do projeto, visitámos três associações de utilizadores de CSA. Isto estabeleceu o ponto de partida para compreender em profundidade a CSA e as necessidades dos seus utilizadores. Tal acabaria por traduzir-se nas decisões tomadas em relação ao sistema de *compansion*. Efetivamente, um sistema que amplie as expressões numa linguagem telegráfica em frases bem construídas em linguagem natural representaria uma melhoria na qualidade de vida dos utilizadores de CSA.

A primeira decisão que tomámos foi quanto ao vocabulário a empregar no sistema. Devido à complexidade que constituiria a construção de raiz de um vocabulário de CSA, decidimos usar um vocabulário já existente, o CACE. Trata-se de um vocabulário básico, com mais de 800 itens, especialmente concebido para atividades da vida diária, e que pode ser integrado em sistemas comerciais de CSA como o Grid 2, o Speaking Dynamically Pro ou o Plaphoons.

Reduzimos o vocabulário inicial a 600 itens e enriquecemo-lo com informação sintática e semântica, tendo em conta os requisitos do analisador e do gerador. Também adicionámos ao conjunto inicial alguns modificadores de frase, que permitiriam uma maior flexibilidade na construção da frase de entrada. Outra informação com que foi possível enriquecer o vocabulário consiste nos padrões que codificámos para cada um dos verbos presentes no vocabulário. Estes padrões descrevem as estruturas das frases que cada um dos sentidos de um mesmo verbo permite ou determina.

Em seguida, foi construído um sistema protótipo para a entrada da linguagem telegráfica. Este sistema não é adequado para um sistema de CSA e não será utilizado na aplicação adaptada ao utilizador final, mas serviu apenas para que fosse

possível alimentar o resto do sistema de *compansion* com o vocabulário e as expressões telegráficas a analisar.

Em seguida, procedeu-se ao desenho conceptual do analisador e à sua implementação. O analisador só aceita um vocabulário reduzido e usa uma gramática controlada, pois não seria possível aceitar todo o tipo de frases presentes na linguagem natural. O analisador extrai os padrões de frase, tendo em conta as palavras de entrada (basicamente, os verbos presentes na entrada) e os seus modificadores. Em seguida, atribui as palavras remanescentes às diversas posições estruturais desses padrões frásicos, os quais correspondem aos papéis semânticos associados aos diversos elementos predicativos e aos complementos de outras palavras. Para isso, o analisador usa principalmente a informação semântica codificada nas palavras e nos padrões verbais. O sistema utiliza apenas informação sintática (ou seja, a ordem das palavras, dependente do idioma) para desambiguar entre as palavras que podem ocupar as mesmas posições, quando a informação semântica não é suficiente. Também usa informação sintática para os substantivos que funcionam como complemento dum outro substantivo, e que o sistema restringe aos substantivos que ocorrem imediatamente a seguir a outros substantivos na entrada. Isso faz com que o analisador seja praticamente independente do idioma. Adaptá-lo para outras línguas, como ao espanhol, envolveria apenas pequenas alterações no código (além da marcação de todo o vocabulário).

Após esta fase, desenvolvemos o gerador, que constrói a frase em linguagem natural a partir da árvore de análise resultante. O gerador é constituído por vários módulos independentes, que progressivamente expandem a árvore de análise de entrada passo a passo. Primeiro, um módulo de ordenação das posições estruturais reordena os componentes da frase de acordo com o tipo de frase. Em seguida, um módulo de ordenação de palavras reordena internamente as palavras em cada uma das diferentes posições estruturais; este módulo acrescenta ainda as preposições e garante que as palavras concordam entre si em género e número. Depois, entra em ação um módulo para o tratamento dos artigos, e que os contrai com os substantivos, se necessário. Este módulo é seguido pelo conjugador dos verbos. Finalmente, um módulo de limpeza lida com os detalhes finais e obtém a frase pronta para a saída.

Após construir o sistema de *compansion*, procedemos à sua avaliação.

Pedimos a três anotadores independentes introduzir um conjunto de 100 frases usando o módulo de entrada de linguagem telegráfica. As 100 frases foram adaptadas a partir de um conjunto de 120 frases que uma das associações que visitámos emprega para ensinar e treinar novos utilizadores de CSA no uso dos seus painéis ou dispositivos de CSA. Os anotadores, em seguida, avaliaram separadamente a saída do analisador e a saída final do gerador. Com isto quisemos avaliar não só se o sistema podia aceitar diferentes entradas para a mesma frase e construir uma saída correta, como também tentámos avaliar os dois componentes principais do sistema, o analisador e o gerador. Pelos números obtidos a partir dos três anotadores independentes, os resultados foram muito promissores: 98,7% das frases geradas pelo sistema foram consideradas corretas.

Resumindo o que conseguimos: construímos um sistema de compressão-expansão (em inglês: *compansion*) para Catalão. Tal tem lugar, obviamente, no âmbito duma gramática controlada, mas o sistema é capaz de produzir frases bastante complexas e, acima de tudo, para as estruturas que aceita, obtém de facto bons resultados. Mais ainda, no contexto da CSA, a utilização de uma gramática controlada não é necessariamente um inconveniente, porque uma gramática mais complexa provavelmente significaria, para o sistema de *compansion*, uma maior taxa de erros e, para os utilizadores finais, um grau de dificuldade mais elevado na introdução das frases. Tendo em mente o contexto final de uso futuro desta aplicação, pensamos que a gramática controlada é mais do que suficiente na maioria dos casos e que o sistema será equilibrado em termos de dificuldade na entrada e de qualidade na saída. Também será uma melhoria em relação ao software de CSA atualmente existente, que ainda produz enunciados sob a forma de linguagem telegráfica. No entanto, assim que for possível testar a aplicação completa, verificar-se-á se a gramática precisa de ser expandida, estando já o sistema concebido para esta possibilidade.

## ACKNOWLEDGMENTS

Thanks to my supervisors, to the teachers and to all the personnel in the faculties that have helped me a lot during these past two years.

Thanks to the AAC associations that I visited for their great assistance and to the annotators for their patience and outstanding work.

Thanks to the Erasmus Mundus Consortium for giving me this opportunity and for all the people that I met during the masters, which have enriched my Erasmus experience.

This project was supported by the European Commission, Education & Training, Erasmus Mundus: EMMC 2008-0083, Erasmus Mundus Masters in NLP & HLT programme.

## TABLE OF CONTENTS

## INDEX OF FIGURES

## INDEX OF TABLES

## LIST OF ABBREVIATIONS

- AAC: Augmentative and Alternative Communication.
- ALS: Amyotrophic lateral sclerosis.
- API: Application Programming Interface.
- ARASAAC: Portal Aragonés de la Comunicación Aumentativa y Alternativa.
- CACE: basic set of pictographic vocabulary for AAC developed by UTAC.
- CCN: Complex Communication Needs.
- *Compansion*: Compression-Expansion. It is a techniques that expands uninflected content words into syntactically and semantically well-formed sentences.
- CPU: Central Processing Unit.
- HLT: Human Language Technology.
- HMM: Hidden Markov Models.
- HTS: HMM-based speech synthesis.
- iOS devices: iPhone, iPad and iPod Touch.
- MS: Multiple sclerosis.
- NLP: Natural Language Processing.
- PCS: Picture Communication Symbol.
- Q&A: Questions and Answers.
- SAPI: Microsoft Speech API.
- SAW: Special Access to Windows.
- SGD: Speech Generating Device.
- SPC: Sistema Pictográfico de Comunicación.
- TALP: Centre de Tecnologies i Aplicacions del Llenguatge i de la Parla.
- TBI: Traumatic Brain Injury.
- UAB: Universitat Autònoma de Barcelona.
- UAlg: Universidade do Algarve.
- UFC: Université de Franche-Comté.
- UPC: Universitat Politècnica de Catalunya.
- USB: Universal Serial Bus.
- UTAC: Unitat de Tècniques Comunicatives de Comunicació of the University of Barcelona.
- VSD: Visual Scene Display.
- wpm: words per minute.

# 1. Overview of the project

## 1.1. Introduction to AAC

Augmentative and Alternative Communication (AAC)[1] [Larranz, 2006] is the set of ways, methods and strategies used by people with certain disabilities that prevent them from communicating normally through natural language or speech. In order to maximise communication, intervention programs are designed to enhance the communication capacities of the user. Residual speech, gestures, communication through graphic signs (pictograms), the use of special communicators or other systems to facilitate access to computers are some of the most common methods for AAC users to communicate. All these interventions also need to take into account the preferences and priorities of the user and his motor, sensory, cognitive, linguistic, psychological and behavioural skills.

Note that people from all ages can have communication impairments, from birth to adulthood. The source of the communication disorders [Kent, 2007] is also vast: cerebral palsy, autism, mental deficiency, aphasia[2] [Kraat, 1990], dysphasia, dementia, multiple sclerosis (MS)[3], amyotrophic lateral sclerosis (ALS), Parkinson, Alzheimer and traumatic brain injury (TBI), among others. While some AAC devices, techniques and strategies require certain language or technical skills and abilities, AAC interventions can help support communication to individuals across all skill levels and ages [Blackstone, 2007].

The main goals of AAC are the following:

- Provide alternative means of communication until speech is fairly restored.
- Provide alternative means of communication for lifetime, when speech is non-existent or severely affected.
- Support the development and restoration of speech and other linguistic skills.

---

[1] What is AAC? http://everyonecommunicates.org/aacintro.html & International Society for Augmentative and Alternative Communication. http://www.isaac-online.org/en/aac/index.html [May 1st, 2011]

[2] AAC-RERC. AAC for Aphasia. http://mcn.educ.psu.edu/dbm/Aphasia/index.htm [May 1st, 2011]

[3] Asociación Española de Esclerosis Múltiple. *Ayudas técnicas para la comunicación.* http://www.aedem.info/portal/index.php?option=com_content&view=article&id=556:ayudas-ticas-para-la-comunicaci&catid=48:noticias-mcas&Itemid=166 [May 1st, 2011]

There is the wrong belief that AAC might hinder the development of a person or the return of his speech. Actually, it is the opposite; in many cases, intelligible speech improves after AAC is introduced to an individual [idem]. Therefore, when an AAC intervention begins, it does not mean that professionals are giving up on speech, contrary to what was thought before research on AAC began.

In turn, communication partners also play a critical role in AAC. First of all, many of the AAC users require a speech language pathologist or a member of their family to set up and customise the AAC device or application of the user. Then, in low-tech communicators [Augé & Escoin, 2003], such as paperboards, the intervention of the partner is crucial, because they play the role of the voice synthesiser in high-tech devices and need to make sure that what they are saying is in fact what the user is pointing and, in other words, what the user wants to say. Usually, primary communication partners play multiple roles on the lives of persons who rely on AAC. Apart from conversational partners, they can be AAC facilitators, technicians of the AAC communicator, trainers of other partners or even caregivers. Therefore, AAC systems need to take into account how to support interactions with a wide variety of communication partners, ranging from experts to people unfamiliar with AAC.

To sum up, AAC technologies allow individuals with complex communication needs[4] (CCN) to expand their social network and to help them fulfil their roles into society as family members, friends or employees at work. Nevertheless, it is important to keep in mind that communication is not just restrained to face-to-face speech, to reading or to regular written communication. Communication on the 21st century involves phone calls, emailing, instant messaging, blogging, twittering and much more. Thus, although new technologies make the technical aspects of AAC easier, the challenge of AAC keeps growing everyday.

## 1.2. Project objectives

The initial goal of the project was to build a free Augmentative and Alternative Communication (AAC) application for Catalan to be used in iOS devices[5] in order to help disabled people with communicative problems in their daily life.

[4] The estimated number of severely speech impaired people living in the European Union is two million. Studies reveal that around 1.5% of the population experience communication disorders (Golinker, 2009).

[5] iOS devices are devices that share iOS, the operating system of Apple for portable devices. At the moment, there are 3 available iOS devices: iPhone, iPad and iPod Touch.

Due to the restricted time available for the development of the project within the timeframe of the masters, we decided to just focus on the development of the core feature of the application, which is the one that will mean a significant improvement over existing systems.

The final application that we expect to build after the masters uses a Picture Communication Symbol (PCS)[6] [Augé & Escoin, 2003] system. The sentences built with this system will be eventually read aloud by a speech synthesizer. As the highlighted feature, true focus of this dissertation project, the application will transform telegraphic language[7] into simple natural language sentences, also known as cogeneration or *compansion*. The final software will also have a pictogram prediction system that will evaluate the frequency of usage of the symbols by the user depending on the previously selected items on the sentence.

Therefore, as this project is enclosed in a bigger project, as both are extremely intertwined, we need to see what AAC is and define the full application to understand the needs and the decisions taken when developing the *compansion* system.

Finally, the full project, as defined above, will only focus on the expressive aspect of AAC without taking into account understanding issues or reception problems that some AAC users can have.

[6] PCS Software. http://www.widgit.com/aboutus.htm & Picture Communication Symbols. http://en.wikipedia.org/wiki/Picture_communication_symbols [May 1st, 2011]

[7] Telegraphic language is characterised by containing only meaningful words [12].

# 2. State of the Art Review

## 2.1. Brief history of AAC

Before 1960, the only population that was eligible for alternative communication strategies were literate individuals, without any other language disorder apart from speech problems, which necessarily had the ability to point. The AAC methods available were limited to typewriters or to boards with the alphabet written on them [Higginbotham, 2007].

During the 60s, professionals realised that they were not obtaining the expected results using traditional methods. That is when symbols were first introduced as an AAC technique [Larranz, 2006]. The focus shifted from speech and language itself, to the ability of communicating regardless of the means. Also, thanks to the electronic innovations, the first dedicated communication devices were created (namely the POSSUM communicator[8]).

Moreover, also in the 60s, scanning [Lesher et al., 1998] was introduced. By pressing a button or a switch, a user could move a cursor from one item (be it a letter, a word, a sentence or a message) to another. This innovation was the key to opening AAC to almost anyone, regardless of the severity of their physical disabilities.

On the 70s, communication systems based on graphical symbols, like Bliss [Augé & Escoin, 2003] or Makaton[9], appeared. Through the combination of these symbols, these systems create a language of their own in a way similar to how single Chinese ideograms combine to form new meanings. Consequently, the user needs to go through a learning process before fully using the system. Despite the learning curve, which is still now being discussed by the AAC professional community, these systems obtained encouraging results (Bliss is still widely used nowadays), which led to a spreading of AAC systems. Also in the 70s, the previously developed communication devices started to be manufactured, thus the AAC industry was born.

During the 80s, manufacturers introduced rate [Garay-Vitoria & Abascal, 2004] enhancement techniques, like abbreviation expansion and linguistic prediction techniques. The appearance of the microprocessor revolutionised mainstream

[8] POSSUM. http://www.possum.co.uk/ [May 1st, 2011]

[9] The Makaton Charity. http://www.makaton.org/about/about.htm [May 1st, 2011]

technology. This change was soon reflected in the type of AAC devices that became available. The first prototypes of AAC computer programs were also developed and more pictographic languages were created. These, like Minspeak[10], relied on multi-meaning drawings, which depend on the order of the sequence in which the drawings appear.

In the social aspect, in some countries, laws that demanded civil rights for disabled people were passed and universal access to communication began to be a public policy issue.

Eventually, in the 90s, new improvements on AAC technologies, such as synthesised speech, dynamic displays, highly accessible symbol sets and smaller and more powerful AAC devices, extended the access to AAC communication devices[11]. Unfortunately, most of these speech-generating devices (SGDs) were very difficult to use and had a steep learning curve, even for family members and AAC facilitators.

## 2.2. Learning process of AAC

The evolution in AAC research reflects the process of learning the use of AAC systems for individuals with CCN. First, communication learning starts with tangible objects, specially adapted games and toys for children. Then, patients move to photographs, which are related to their environment, and afterwards to images. At this stage, visual scene displays[12] (VSDs) can be introduced to users. VSDs are specially indicated to children [Reichle & Drager, 2010] and to users with certain types of aphasia [Peña-Casanova, 1991].

Capable individuals continue by integrating the images, pictograms or graphic symbols into communication boards that usually display these images on a grid ordered by semantic groups or by syntactic categories (like nouns, verbs, adjectives, etc.). In most modern PCS systems, to better distinguish these categories, each of them has a colour associated, be it the background colour of the image or the colour of the frame of the image on the grid. Basically, the aforementioned VSDs, instead of

[10] Introducción y enseñanza del sistema Minspeak de comunicación aumentativa. http://www.esaac.org/descargas/GuiaMinspeak.pdf&pli=1 [May 1st, 2011]

[11] Augmentative & Alternative Communication Devices. http://www.youtube.com/watch?v=Eb_URYj_L_k [May 1st, 2011]

[12] AAC for Aphasia: A Review of Visual Scenes Display Project. http://mcn.educ.psu.edu/dbm/Aphasia/index.htm [May 1st, 2011]

placing pictographic symbols, use photos of experiences, situations or contexts familiar to the individual. The colour caption usually used is the following:

- Yellow for nouns, proper nouns and pronouns that refer to people.
- Green for verbs.
- Orange for common nouns.
- Pink for vocabulary or expressions related to social relations.
- Blue for adjectives.
- White for numerals, colours, days of the week, holidays (Christmas, Easter, etc.), modifiers and other miscellaneous words and expressions.

Communication boards can be paper or carton made boards or even electronic boards with their own sound output. Some of these basic communicators have pre-recorded sentences that are accessed through a combination of pictograms. However, the number of sentences is limited and, although the device is able to output proper natural language sentences, which is a positive aspect that we will later discuss, the system does not let the user create new sentences, which is also a big drawback.

Finally, the last step is the transition from static boards, be them manual or electronic, to dynamic boards, which allow to drastically increase the amount of vocabulary available to the user. Most of these dynamic boards are part of AAC software developed for specific AAC electronic devices[13] or for regular computers or other portable devices available for the general public. The complexity of electronic communication devices and their AAC software varies widely based on the needs and the cognitive abilities of the user. These devices can have fully synthesised voices, contrary to the previously mentioned electronic boards that just had prerecorded words or sentences. Specific means of accessing these systems are adapted to the needs of each individual and may involve direct use of the keys or of the touch screens, use of head-mounted pointers, use of gaze-tracking technologies or use of external switches to control the devices[14].

[13] BJ-Adaptaciones. Software de comunicación. http://www.bj-adaptaciones.com/catalogo/software [May 1st, 2011]

[14] BJ-Adaptaciones. http://www.bj-adaptaciones.com [May 1st, 2011]

## 2.3. Current State of AAC

At the moment, the leading AAC commercial software, which can be adapted to many of the individuals with CCN, is a software for Windows-based computers called The Grid 2[15]. The Grid 2 provides a fully customizable dynamic board interface and is available in many languages, including English, Spanish, French or Portuguese. The user can configure each pictogram of the board so that it represents a single letter, a word, an action (i.e. erase the previous selected character) or so that it represents a full set of pictograms and leads to another board. This way, the user or the facilitator can choose the vocabulary according to the needs of the user and embed boards into boards and freely organise them. However, the synthesised output of the sentences that The Grid 2 offers does not produce natural language sentences, in other words, if the user utilises pictogram-based communication and uses a synthetic or telegraphic language, the system does not transform this telegraphic language into properly constructed sentences.

On the other hand, another leading software, far less spread, Speaking Dynamically Pro[16], gives the possibility of language expansion, although it has to be completely programmed by the user and, overall, is not as user friendly as The Grid 2. This software, apart from incorporating all the main characteristics of The Grid 2, allows the user to give features to each pictogram (for example, gender or tense) and associate actions to them. It even permits to set features depending on the features of the other pictograms on the same sentence. Every time that a certain pictogram is used, the software checks its features and its dependencies and applies the programmed actions (i.e. according to the gender, adding the corresponding article, in romance languages, or according to the tense specified by a temporal expression, inflecting the verb).

Both programs are accessible using a wide array of external devices, ranging from regular keyboards and mouse to external switches, head or eye-pointer and any other devices that can substitute the functions of the mouse. Moreover, both systems incorporate word prediction, for individuals that use keyboards, which communicate through regular written language, be it physical keyboards or on-screen keyboards.

---

[15] Sensory Software – The Grid 2. http://www.sensorysoftware.com/thegrid2.html [May 1st, 2011]

[16] Mayer-Johnson. Speaking Dynamically Pro. http://www.mayer-johnson.com/boardmaker-with-speaking-dynamically-pro-v-6 [May 1st, 2011]

Another important aspect of AAC that has greatly improved during this last decade are voice synthesisers. Whilst voice synthesisers used to sound robotic-like as they completely neglected prosody and sometimes were hardly understandable (and even less by people with language disorders), nowadays the most advanced ones feel fluent and natural. Synthesising methods evolved from diphone voices, to HTS voices, which use Hidden Markov Models (HMM), to the more natural clunit voices [Bonafonte et al., 2009], which are based on concatenative speech synthesis. Clunit voices sound more natural than HTS. On the other hand, HTS voices are usually smoother and more stable, while clunit voices can produce rather frequent concatenation errors. HTS voices also require far less memory and CPU resources.

Some of the leading companies that provide voice synthesising services are Acapela[17] and Loquendo[18]. There are also remarkably good, open-source, synthetic voices, such as the one available for the Festival engine[19], created by the Centre for Speech Technology Research from the University of Edinburgh.

Concerning pictogram-based communication, apart from the aforementioned symbol languages, like Bliss or Minspeak, other, more straightforward, and intuitive pictogram systems are available, like the widely spread SPC (Sistema Pictográfico de Comunicación) system [Augé & Escoin, 2003], which can use different symbol sets like Widgit-Rebus[20] or the open source ARASAAC[21] set. In SPC, each pictogram or drawing represents a single word, whose meaning is much easier to guess without nearly any learning process than in highly symbolic languages like Bliss.

Finally, regarding the AAC scene in Catalan, during the 20th century, before the appearance of the first computer-based AAC systems, boards and other AAC methodology were primarily developed for Spanish and could be easily ported to Catalan. However, since then research progressed much slower in Catalan, as most of the effort put in developing AAC software was focused in Spanish, until the

[17] Acapela. http://www.acapela-group.com/ [May 1st, 2011]

[18] Loquendo. http://www.loquendo.com/ [May 1st, 2011]

[19] The Festival Speech Synthesis System. http://www.cstr.ed.ac.uk/projects/festival/ [May 1st, 2011]

[20] Widgit Rebus. http://www.widgit.com/aboutus.htm [May 1st, 2011]

[21] Portal Aragonés de la Comunicación Aumentativa y Alternativa. http://www.catedu.es/arasaac/ [May 1st, 2011]

appearance of Plaphoons[22] a free AAC application for Windows, available both in Catalan and Spanish. Plaphoons allows for embedded boards like The Grid 2 or Speaking Dynamically Pro, but lacks word prediction or any other advanced linguistic features, although it allows external switches, head-pointers and similar peripherals to access it. Since late 2009, The Grid 2 and Speaking Dynamically also have basic support, in its non-linguistic features, for Catalan. Both systems use the SAPI technology[23] for Windows, which allows using any synthesised voice in any language available for the operating system. As the open source Festival voices and other commercial voices from Loquendo and Acapela are already available for Catalan, both programs can synthesise text in Catalan.

## 2.4. Future of AAC

While in the past decade AAC interventions have mostly focused on computers, on software for laptops and for specific AAC portable devices, such as Hermes[24] or DynaVox[25], with the fast growth of tablets in the last couple of years, the tendency has changed. Multi-modal devices that are able to perform different tasks (mailing, web browsing, agenda, calculator, accessing social networks, calling, text messaging, gaming, online shopping, listening to music, etc.) are taking over the world. As people with CCN should not be left apart, AAC devices need to be able to perform all these tasks and be compatible with mainstream technologies. That is why computers, although less portable than specific portable AAC devices, were prefered over the latter. But with the sudden rise of tablets and multi-purpose cellphones the picture has changed.

Nowadays, portable devices are gaining ground as AAC technologies for individuals with minimal to mild physical disabilities (i.e. people with Down syndrome, aphasia, some kinds of cerebral palsy, autism, etc.) who want extremely portable devices, with multiple functionalities and a trend factor [Blackstone, 2009]. As time goes by, some peripherals, such as bases to attach devices to a wheelchair, will make them more suitable for users with higher levels of disability, in the same way that it was with desktop computers and laptops. Sound and speech technologies offer natural

[22] VideoFAQ (tutoriales) sobre Plaphoons. http://www.xtec.cat/~jlagares/IndexVideotutorialPlaphoons.htm [May 1st, 2011]

[23] Wikipedia. Microsoft Speech API. http://en.wikipedia.org/wiki/Microsoft_Speech_API [May 1st, 2011]

[24] Hermes. http://www.bj-adaptaciones.com/catalogo/software/software-comunicacion/bj-hermes-pda.html [May 1st, 2011]

[25] DynaVox. http://www.dynavoxtech.com/ [May 1st, 2011]

sounding and amplification, which makes it possible to hear speech generating devices in outdoor conditions, such as classrooms, parties or meetings.

Laptops are too frail for many AAC users, some of whom do not have good motor control. They are not very usable outdoors, where sunlight, weight and short battery life are huge problems. Desktop computers are still widely used too as AAC communicators, but they are located in specific places and can only be accessed in adequate circumstances. Furthermore, in most ocasions, desktop computers are placed next to the wall, making face-to-face communication even more difficult. People with CCN do not want to be tethered to a wall or attached by wires [Escoin, 2006].

The key to solving all these issues seems to lie in tablets. A good sign in this direction is that tablets like iPad already offer multiple accessibility options for diabled users, like VoiceOver[26], that can also benefit future AAC users. However, tablets still do not possess the processing capabilities of desktop computers nor laptops, and do not allow for fully-fledged AAC applications to be developed for them yet. Some basic AAC applications for smartphones and tablets have already appeared, such as Proloquo2Go[27] (for iOS devices) or TapToTalk[28] (for iOS devices and Android[29]).

However, expressing basic needs and exchanging greetings, while it is an important step to functional use of SGDs, true communication requires that the user has means to generate language at any time, anywhere and with anyone. Communication is not just the goal of AAC interventions in itself, but rather a way to many ends. To participate actively in the community or inside the family, to pursue personal interests and goals, allowing the user to lead a high quality lifestyle and achieve self realisation, this is the true purpose of modern AAC [Blackstone, 2009].

In terms of scientific research, applying effective natural language processing (NLP) techniques in AAC for improving the message output has still a long way to go [Copestake, 1997]. Studies reveal that the average rate of AAC systems is between 5 and 20 words per minute (wpm). Even the fastest utterance-based devices only

[26] Apple. VoiceOver. http://www.apple.com/accessibility/voiceover/ [May 1st, 2011]

[27] Proloquo2Go. http://www.proloquo2go.com/ [May 1st, 2011]

[28] TapToTalk. http://www.taptotalk.com/ [May 1st, 2011]

[29] Android. http://www.android.com/ [May 1st, 2011]

reach a maximum of 60 wpm, which is far less than normal conversational rates, which are around 150 to 200 wpm [idem]. This results bring to light the following question: Is the rate of communication overrated? Certainly, normal rates are impossible to achieve in AAC. However, more than trying to increase rates in order to improve communication speed, it is a matter of reducing the number of keystrokes or the amount of gestures that AAC users need to communicate, consequently reducing their overall effort.

While word prediction [Garay-Vitoria & Abascal, 2004] has indeed achieved good results in terms of saving keystrokes, other strategies like iconographic prediction have not been deeply explored. Moreover, despite saving keystrokes, the way in which predicted words are displayed, sometimes cause more mental effort for AAC users and it can make them to become tired faster. That is why new ways of presenting the information need to be explored.

Other interesting AAC areas that need to be further developed would be:

- Gesture recognition and eye-laser systems [Blackstone, 2009]: while head-pointers and external switches are widespread, other computer accessing methods like mild eye-laser and gesture recognition need to be researched. This area presents a huge challenge: how will it be possible to tell apart extraneous movements, which are quite common among AAC users, from meaningful movements?
- Better techniques to transform telegraphic language into natural language. Up until now research in English [Pennington & McCoy, 1998], Greek [Karberis & Kouroupetroglou, 2002], French [Vaillant, 1997], Japanese and other languages has been done, although it focused on simple sentences rather than on complex sentences, like subordinate constructions.
- In the field of voice synthesisers, future research will focus on the possibility that each user could parameterise the voice of the program creating a custom and unique synthetic voice of their own [Blackstone, 2009]. Research also needs to improve on the representation of emotions (yelling, singing, talking to a pet, etc.).
- Dysarthric speech recognition [Escoin, 2006], so that AAC devices could act as interpreters for people with dysarthria.
- Research new ways to better distribute AAC content into displays, in order to facilitate navigation and enhance the rate of communication. It is also

necessary to develop even more intuitive and user-friendly applications that can help reduce the learning curve of AAC systems.

- Further investigate the interaction between AAC users and their communication partner, in order to develop new ways for AAC systems that can help maintain the attention of the partner. A possibility would be for the system to say a sentence similar to “Wait a moment, please” every time that the users begins to input a sentence [Copestake and Flickinger, 1998].
- Applying state-of-the-art research to commercial AAC software available to the general public. Usually, some of the most advanced features of AAC are just implemented in prototypes developed specifically for research or for specific individuals [Pennington & McCoy, 1998].

## 2.5. Dependency parsing review

As the *compansion* system that we have built is based on the concept of dependency grammar, we thought that it would be interesting to add a small literature review on dependency parsing to this dissertation.

Dependency parsing is the computational implementation of syntactic analysis using dependency representations. In dependency parsing, as opposed to parsers based of constituency analysis, the relation between the theoretical frameworks and the computational implementation can be quite shallow. This is probably due to the low degree of formalisation of dependency grammar theories.
Also, when talking about dependency parsing, usually two different approaches are distinguished: *grammar-driven* parsing and *data-driven* parsing. In this review, we will only talk about grammar-driven parsing, which is the approach that we have taken.

### 2.5.1. Grammar-driven dependency parsing

The first works on dependency parsing were done by Hays (1964) and Gaifman (1965). Their approaches were close to the formal theories of dependency grammar. Most of the basic notions of dependency grammar (Nivre, 2005) were present, such as the single-head constraint, the representation resulting in a rooted tree and the applicability of the projectivity constraint, among others. These first parsers did not use any dependency types to classify dependency relations, thus dependencies remained unlabelled.

The results that Gaifman (1965) obtained, which closely related his dependency system to context-free grammars, discouraged the further study of dependency

grammar for parsing. However, Järvinen and Tapanainen (1998) disclosed that the conclusion that dependency grammar is only a small variant of context-free grammar reached by Gaifman was erroneous.

Following the framework that Gaifman and Hayes first proposed for dependency parsing, the algorithms that appeared used dynamic programming in the same way that parsing algorithms for context-free grammars did. Most of the frameworks that later appeared also implemented the notion of projectivity, except for some parsers that introduced non-projective structures after a post-processing step (Sleator and Temperley, 1991). Most of these frameworks can also be included under the concept of bilexical grammar introduced by Eisner (2000). The basic parsing algorithm proposed by Eisner also uses dynamic programming.

The second principal tradition in grammar-driven dependency parsing is based on the method of eliminative parsing. In this method, sentences are analysed by sequentially eliminating representations that do not fit constrains until only a valid representation remains. As we can see, this notion is closely related to the notion of constraint grammar.

The first dependency parsers using this idea came from the Constraint Grammar framework (Karlsson, 1990). Afterwards, Maruyama (1990) extended the idea using a system that tagged dependencies with both a syntactic label and an identifier for the head node. This type of representation for dependencies is fundamental for many approaches to dependency parsing, as it reduces the parsing problem to a problem of classification or tagging.

In the eliminative system, parsing is a constraint satisfaction problem, where a good analysis is the one that does not break any of the constraints of the grammar. In general, this type of constrain satisfaction problem is NP complete. In this type of problems a given solution can be quickly verified, but, on the other hand, there is not a known efficient way to find a solution. The time to solve these problems rapidly increases as the size of the problem grows, thus, when implementing algorithms that solve this kind of problems, you need to be cautious, so that it is fast enough and does not take too much time.

In more recent approaches, like the TDG framework (Duchier 2003) faces this problem using constraint programming, which ensures the finding of a solution to the

parsing of a given input in a reasonable computing time. The TGD framework also introduces several levels of representation in dependency parsing (notion that we already saw), exposing that a single constrain can point to different levels (i.e. syntactic and semantic) at the same time (Duchier and Debusmann, 2001). This view is also extended in the Extensible Dependency Grammar framework (XDG) (Debusmann et al., 2004) where many levels or dimensions can be defined in the grammar.

The two main problems that the parsing frameworks that we have seen up until now are: first, there might not be an analysis, for a given input, that fits all the constraints in the grammar and, second, there might be more than one analysis that fits all the constrains, in other words, there can be a disambiguation problems. To face both issues, the notion of weighted constraints appeared (Menzel and Schröder, 1998). In this approach, a weight that indicates how serious is the violation of a given constraint, is assigned to each constraint. Therefore, the best analysis is the one for which the sum of violated constraints is minimised.

Finally, there is a third grammar-driven parsing tradition that combines dependency grammar with a deterministic parsing strategy. The basic strategy of this tradition is defined by Covington (2001) in the following way:

> *"Accept words one by one starting at the beginning of the sentence, and try linking each word as head or dependent of every previous word."*

This strategy is compatible with many different types of dependency grammar. The only thing that is required is for the grammar to define a function that for any two words it returns a Boolean, true if the first word can be the head of the second words or false otherwise. Covington (2001) demonstrates that this method can be used to obtain dependency representations that satisfy conditions like uniqueness (having a single head for each node) and projectivity by adding pertinent constraints on the linking process. Covington (1990a,b, 1994) also showed that this method can be adapted to languages with different types of word order, be it free, flexible or rigid.

To conclude with grammar-driven dependency parsing, we have seen three different traditions. The first one is more based on the formalisation of dependency grammar theories that are mostly restricted to projective dependency representations. The first

tradition uses dynamic programming algorithms to implement the parsers. The second one is based on the formalisation of constrains, which do not need to be restricted to projective structures, and the parsing method uses a successive eliminative approach. Finally, the third one is based on the combination of dependency grammar with deterministic parsing strategies and the parsing method is the one described by the quote of Covington (2001). This last strategy is the one that is most similar to the one that we have taken for the the parser of the *companion* system (see the Methodology section).

# 3. Motivation

To the best of our knowledge, no AAC application that expands telegraphic language into natural language still exists for Catalan or even for Spanish. Persons that are not familiar with people with CCN might not take AAC users seriously, as they can infer from their telegraphic speech that they have some kind of intellectual disability. In turn, people that have full or nearly intact linguistic competence, but that have speech impairments, also feel frustrated when the AAC systems that they use cannot produce natural language. Because of this, taking into account the social and psychological benefits of being able to communicate using natural language, both for users and for their families and friends, there is a need for a basic and free AAC application in Catalan that expands telegraphic language into natural language. Fulfilling this need (at least partially) is the aim of this dissertation project.

Moreover, the most widespread free AAC application in Catalan, the previously mentioned Plaphoons, apart from lacking some interesting features for an AAC application, like easily navigable boards or a pictogram prediction system, is not available for portable devices, which are the present and future of AAC. That is why there is a need for a portable and free AAC system in Catalan, which also includes more state-of-the-art features.

Another important aspect is making the application accessible for as many users as possible[30]. That is why we decided to build the *compansion* system using web-oriented programming languages and technologies, to make it accessible by any device that has Internet connection, be it desktop computers, laptops, tablets or smartphones.

## 3.1. Potential Users

Design features vary for different groups of individuals with congenital or acquired disabilities and CCN. For example, individuals with ALS do not typically have

[30] Eventually, the application is meant to be distributed. In order to increase the diffusion of the software, using the name Plaphoons, has been suggested by the Departament d'Educació de la Generalitat de Catalunya (State Department of Education of Catalonia). This option will be discussed in later stages of the project.

language disorders, while people with aphasia or dementia often have severe language impairments.

The application described in this project is aimed at individuals presenting some speech disorders (certain types of aphasia, disartria or other motor speech impairments, ALS, MS, specific cases of cerebral palsy, TBI, deaf-muteness, etc.), but that keep intact (or lightly affected) their linguistic competence. For some of these potential users, the application will only help them in some stages of their condition. For example, for individuals with TBI, it can help them to communicate their basic needs during their rehabilitation process to recover speech.

In principle, the application will not be addressed to children, as it will require some linguistic knowledge to be fully used. Moreover, children also have special needs [Reichle & Drager, 2010] regarding the user interface and the distribution of the pictograms. However, children that are already familiar with VSDs or low-tech AAC devices could also use it as part of their learning process with alternative communication. It can also help them improve their language skills, as the application will turn telegraphic utterances into basic natural language sentences.

Granting software access to the application to users with severe motor impairments through alternative means (like mouses, switches, external keyboards or other peripherals) will not be among the main objectives of the project. Nevertheless, while building the final application, we will make sure that the interface can be adapted to a wide array of peripherals implementing scanning techniques and key mapping. Designing and building potential AAC peripherals for iPhone and mostly for iPad will be up to third parties. Some of these already exist[31].

Finally, another unexpected group of users could be second language learners of Catalan, which, at early stages of their learning process, could use the application to learn how to build simple natural language sentences in Catalan from the selected keywords in the input of the system. Nevertheless, the system would not need to be adapted for them, as, in principle, they would not have any special needs.

[31] RJ Cooper & Associates, Inc. http://www.rjcooper.com/ [April 15th, 2012]

# 4. Methodology

Apart from documenting ourselves through reading papers and reviews on AAC, we also met with three associations for disabled people[32] that use AAC strategies and devices and with the person in charge of accessibility for disabled students of the State Department of Education of Catalonia (Departament d'Educació de la Generalitat de Catalunya). With their help and advice, we learned how pictogram-based communication worked and, with the experience acquired, we were able to better determine the needs of the final application, to limit the types of sentence structures that the *compansion* system would accept and to define the methodology to follow in order to build it.

Before discussing the methodology followed to build the *compansion* system, we would like to better define our project and its parts.

In short, our dissertation project has to do with the transformation of telegraphic language[33] used in pictogram-based AAC into natural language utterances in Catalan. The telegraphic language that will need to be expanded will result from the input of pictograms, which only contain drawings for meaningful words (in other words: verbs, nouns, adjectives, a small set of adverbs, possessive determiners, quantifiers and some set expressions, plus sentence modifiers like tense modifiers, order or request modifiers, word modifiers (plural modifier), etc.).

The main characteristics of this telegraphic language, apart from only containing content words (and some modifiers), as we have just seen, are the following:

- Words can appear in any order. Thus, it is a free word order language.
- There can also be reduction of some content words (like the subject of the sentence).

An example input could be:

---

[32] Centre d'Educació Especial Pont del Dragó http://www.bcn.es/pontdeldrago/ca/index.html & Prodiscapacitats Fundació Privada de Terrassa. http://www.prodis.cat/ & Associació Pro-Disminuïts Físics i Psíquics de Sant Cugat del Vallès. http://www.asdisantcugat.org/ [April 15th, 2012]

[33] Telegrpahic language is characterised by containing only meaningful words (Karberis and Kouroupetroglou, 2002)

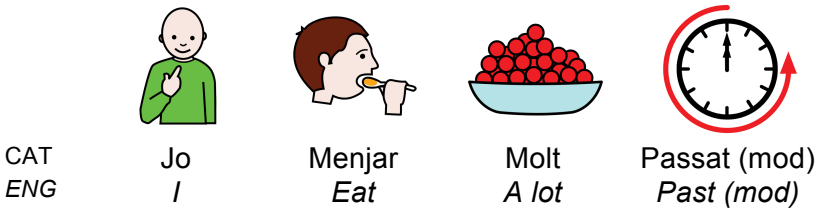



**Figure 1: Example of input with a tense modifier.**

The input in Figure 1 would expand to the sentence “He menjat molt” (*I ate a lot*). Therefore, as just seen, the first part of the *compansion* system needs to be an input system.

In this project, due to time constraints, we decided to create an alternative input system that substitutes the sets of pictograms, for lists of words (Figure 2). As each pictogram translates into a single word (pictograms represent meaning, so if a word can have two senses, it will have two different pictograms related to it), for the rest of the *compansion* system, it does not make any difference. However, this input system is not suited for the end users of the final application, so it is a provisional system with the only aim to feed and test the *compansion* system. Discussion on the final input system that will be left for future work can be found in section 4.6.

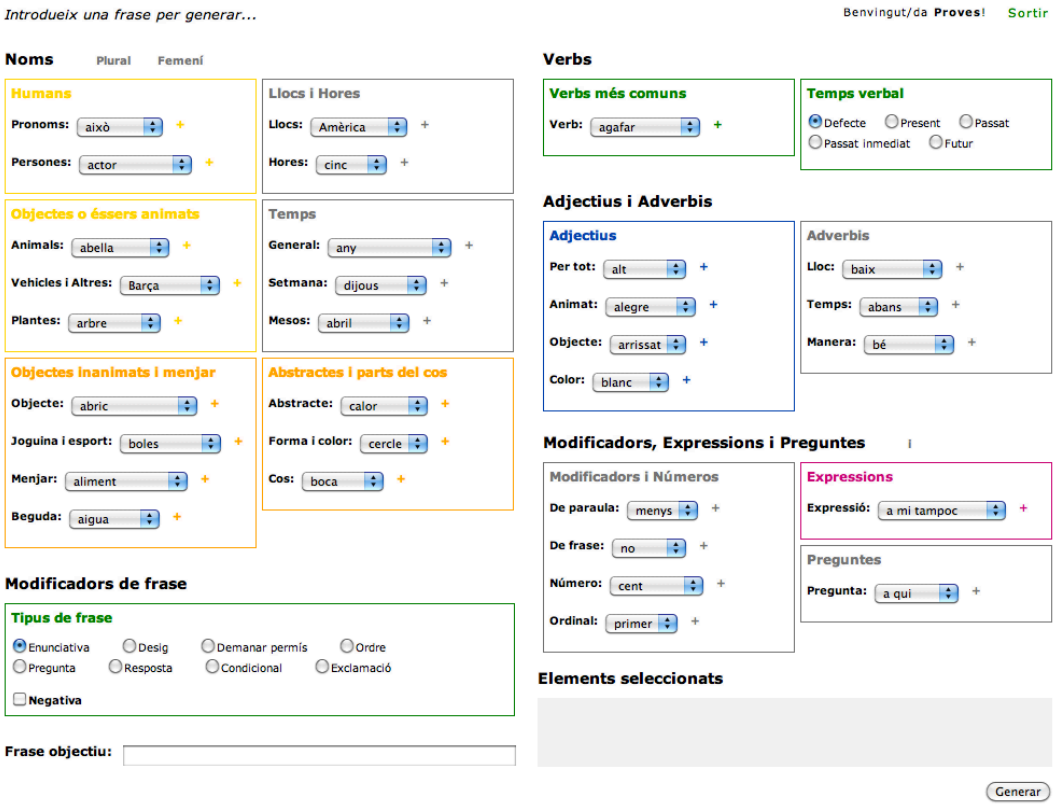



**Figure 2: Screenshot of the provisional input system. Lists of words are grouped into different categories.**

After the input system, the core of the *compansion* system comes into play. First of all, there is the parser, which takes the input words from the input system, which are in the form of telegraphic language, and tries to assign them their correct function in the sentence. Then, the generator takes the parse tree built by the parser, plus the sentence modifiers that might have been input as well, and, using a generation model, expands the sentence transforming it into a natural language sentence (Figure 3).

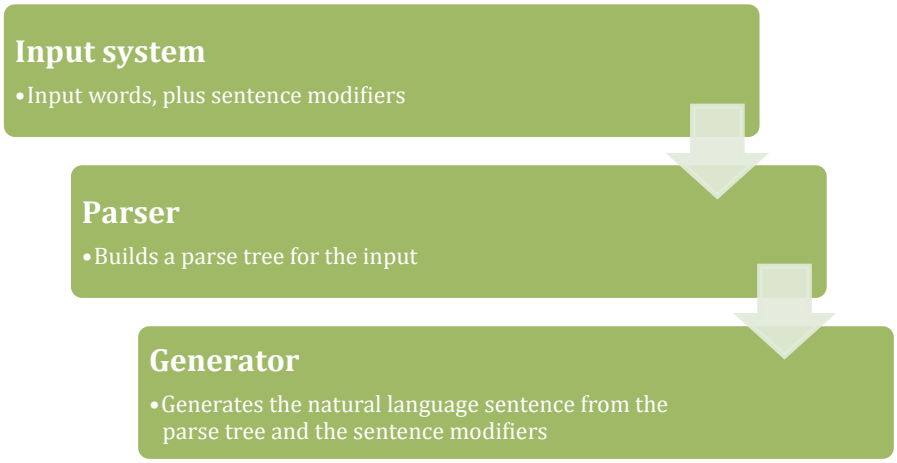



**Figure 3: Components of the *compansion* system.**

In order to expand the telegraphic language into natural language, the parser uses a semantic approach similar to the one that can be found in Pennington and McCoy (1998). The basic idea is that the verb is the centrepiece of the sentence, which perfectly adapts to dependency grammar where the verb is the root, and the rest of the complements of the sentence that need to be filled are the semantic roles (or slots) that that verb accepts. The semantic roles accepted by each verb of the lexicon are described in a set of patterns. The patterns also have default values that are used in case there is a reduction of one of the mandatory slots for a given verb (Figure 4).

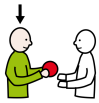
| | | |
|---|---|---|
| |  |  |
| *CAT* | Donar | poma |
| *ENG* | To give | apple |
| *CAT* | Em pots donar una poma, si us plau. | |
| *ENG* | Can you give me an apple, please. | |

**Figure 4: Example of a sentence with a request modifier using telegraphic language. In this sentence there is a reduction of the subject and a default subject for a request is used instead.**

Also, as trying to accept all the constructions allowed in Catalan grammar would be too ambitious for the project, we decided to narrow down the target constructions to

basic structures using a controlled grammar. Here are the main restrictions of the system, along with some examples:

1. The grammar only allows one type of subordinate clauses in the sentences. It only allows for verbs that directly depend on other verbs and does not allow any other type of subordinate clauses, like relative clauses or subordinated clauses leaded by a subordinate conjunction (Figure 5).
    a. In turn, a verb that already depends on another verb cannot be the head of another verb. Therefore, the system only allows for a maximum of two verbs in a sentence (unless some sentence modifiers, like "Desig" (*Wish* modifier) are used.

| | | |
|---|---|---|
| Estic cansat de jugar. | I'm tired of playing. | Accepted. |
| Ajuda'm a baixar les escales. | Help me go down the stairs. | Accepted. |
| M'agrada cantar. | I like singing. | Accepted. |
| Espero que vinguis a veure la pel·lícula. | I hope that you come to watch the movie. | Not accepted (there are three verbs in the sentence). |
| Vull anar a comprar. | I want to go shopping. | Accepted (only if the modifier *Wish* is used, because there are 3 verbs in the sentence). |
| L'home, que passeja el gos, és simpatic. | The man that walks the dog is very nice. | Not accepted (it's a relative clause). |
| Era intel·ligent, però va cometre un error. | He was smart, but made a mistake. | Not accepted (subordinate conjunctions are not supported). |

**Figure 5: Examples of accepted and not accepted sentences by the parser.**

2. Apart from subordination, the system does not accept coordination of sentences either.
    a. Nevertheless, the parser accepts coordination between two nouns or between two adjectives (Figure 6).

| | | |
|---|---|---|
| La Maria és llesta i alegre. | Mary is smart and cheerful. | Accepted. |
| L'home i la dona fan pastissos. | The man and the woman make cakes. | Accepted. |
| Ell canta i balla. | He sings and dances. | Not accepted. |

**Figure 6: Examples of accepted and not accepted sentences regarding coordination by the parser.**

3. Nouns that are the head of a slot (e.g. the head of the subject or the head of the theme in the sentence) can have the following complements/modifiers (Figure 7):
    a. Another noun, just one (or two coordinated nouns).
        - A noun that is complementing another noun cannot have any more complements (but can be modified by for adjectives).
    b. An adjective, just one (or two coordinated adjectives).
    c. Quantifiers. There are the following quantifiers in the system: "molt", "poc", "més", "menys" (*very/much*, *a few/little*, *more*, *less*). Two quantifiers can appear together: "molt més alt" (*much taller*).
    d. Numerals[34].
    e. A possessive.
    f. A locative adverb (if the noun is the head of a locative slot).
    g. All the previous at the same time (except for numerals and possessives that the system still does not handle if they are together and except quantifiers, numerals or possessives that cannot go with adverbs in our system).
4. Adjectives, Adverbs and Quantifiers can have the following complements:
    a. Quantifiers (any number of them).
5. Verbs can have sentence modifiers, like the negative particle "not".
6. Pronouns cannot have any complements.

| | | |
|---|---|---|
| L'home de ferro. | The man of iron. | Accepted. |
| El ninot de neu i gel. | The doll of snow (snowman) and ice. | Accepted. |
| Moltes més pomes. | Many more apples. | Accepted. |
| La casa blanca de pedra. | The white house of stone. | Accepted. |
| La casa de pedra blanca. | The house of white stone. | Accepted. |
| Tres dones altes. | Three tall women. | Accepted. |
| La meva germana més alta. | My more tall (tallest) sister. | Accepted. |
| A sota la taula. | Under the table. | Accepted. |
| No vinguis. | Don't come. | Accepted. |
| Els meus tres germans. | My three brothers. | Not accepted. |
| A sota la meva taula. | Under my table. | Not accepted. |

**Figure 7: Examples of accepted complements for nouns, verbs and adjectives.**

[34] Although they are also modifiers, as they are treated differently by the parser, that is why we put them in two separate categories.

7. Expressions are thought to be the only element in the input, as they alone constitute a sentence on its own. Only a few expressions, like “Si us plau” (*Please*), can appear along with other words in the input.

We think that these constraints are not a big limitation to the system as most sentences built using AAC devices have very few words, so complex constructions are extremely rare and are usually split into simpler sentences.

In turn, the vocabulary for the system is also constraint, as we will use a vocabulary selection specially designed for basic AAC communication by the University of Barcelona, named CACE[35].

It is also important to mention that the *compansion* system assumes that the input will have a correct parse, in other words, that the words that the user entered intend to build a correct, both syntactically and semantically, sentence. As a result, the system does not check for possible input errors in the sentence.
Having seen the basics of the system, lets start defining all its elements in detail.

## 4.1. Vocabulary

The task of selecting an adequate vocabulary for an AAC application is a challenging one, as many different aspects are not trivial or might not easily come to your mind if you are not very familiar with the subject. Moreover, a system with inadequate vocabulary or featuring too many lacunae will frustrate or dismay its users [Youngseo, Eunsil et al., 2006].

Initially, the application had to deal with general greetings, such as introducing oneself, and another daily activities such as shopping or going to a restaurant. In order to select one of these daily life activities, we had several meetings with associations for disabled people around Barcelona that used different kinds of AAC systems. We realised that the interest for each activity highly depends on the preferences of the user and their daily routine, which can widely vary depending on their disability. Some other important external factors should also be taken into account. For example, people with CCN are at high risk of suffering some kind of

[35] UTAC-CACE. http://www.utac.cat/noticies/utac-cacejadisponibleperadescarregarenquatreversions.
A downloadable version of the vocabulary in Catalan can be found at:
http://www.utac.cat/descarregues/L%C3%A8xicCACE.pdf?attredirects=0&d=1 [April 15th, 2012]

abuse, victimisation or crime. Consequently, AAC systems ought to have the vocabulary needed to face these situations.

As selecting an adequate vocabulary in a proper way would need a detailed study of word and sentence usage of people with CCN in different contexts, we have decided to follow the recommendations from the associations that we met and to use an already available set of specific vocabulary for Catalan AAC users named CACE[36] [Soro et al., 2007]. In fact, CACE is a set of ready-to-use boards that conform a pictographic communicator in Catalan and Spanish, which can be integrated into several types of AAC software for computers (The Grid 2, Boardmaker, which in turn integrates with Speaking Dynamically, Plaphoons and SAW[37]).

The most important characteristics of the pictographic communicator CACE, built by UTAC (Unitat de Tècniques Augmentatives de Comunicació), a specialised group in AAC techniques from the University of Barcelona, are the following:

- CACE includes a proposal of basic vocabulary[38] (853 vocabulary items and 132 phrases) that needs to be customised for each user.
- It is organised in categories, 20 in total, in several pages and navigable boards.
- It also includes morphological modifiers (tense modifiers, possessives, etc.).
- It can be extended, while maintaining its general structure.

For our project, as our objective is not to build the full application, we decided to reduce a little bit more the vocabulary present in CACE. In appendix A we can see a list with all the vocabulary in CACE. Words highlighted in green are the ones present in our system.

Mainly, what we have reduced is the number of verbs available (see Table 1). Our aim in this project is to make a functional system that correctly expands telegraphic language into natural language with the vocabulary available. Afterwards, increasing

---

[36] UTAC-CACE. http://www.utac.cat/noticies/utac-cacejadisponibleperadescarregarenquatreversions [April 15th, 2012]

[37] Special Access to Windows. http://informaticaparaeducacionespecial.blogspot.com/2007/07/programa-saw-o-cmo-emular-un-teclado.html [April 15th, 2012]

[38] List of vocabulary of CACE in Catalan. http://www.utac.cat/descarregues/L%C3%A8xicCACE.pdf?attredirects=0&d=1 [April 15th, 2012]

this vocabulary should not be too problematic, as it is just a matter of adding all the related information necessary to each word type, as we will see below.

The decision to reduce the amount of verbs was taken because they are the type of word that takes longer to annotate. For each verb, several patterns representing each of the most common senses or usages of the verb in Catalan have to be described. Each pattern tells which semantic roles are mandatory for that verb sense, which are optional, which type of word and class within that type of word is the ideal head for the slots, default values for mandatory slots, etc. More information on verb patterns can be found in the description of the parser below or in appendix B, where all the patterns of the system can be reviewed.

| Type | # | Total in CACE |
|---|---|---|
| Verbs | 41 | 88 |
| Patterns | 76 | - |
| Nouns (including pronouns) | 470 | 571 |
| Adjectives | 72 | 95 |
| Adverbs | 21 | 21 |
| Set Expressions | 39 | 39 |
| Question particles | 11 | 11 |
| Modifiers (possessives, quantifiers, etc.) | 15 | 15 |
| Sentence Modifiers (type of sentence, tense, etc.) | 14 | 7 |
| **Total words** | **669** | **840** |

**Table 1: Amount of vocabulary and patterns in the *compansion* system.**

For nouns, the following information (features) needs to be annotated:

- Gender: if it is masculine or feminine.
- Number: if it is in singular or plural.
- If it is countable or uncountable.
- Which is the most common article that precedes it (definite, indefinite or no article).
- Its plural form, if it has one.
- Its feminine form, if it has one.
- All the semantic classes to which it belongs (see Figure 8). These classes serve to specify which class of noun better fits a slot and to see which

complements (other nouns, adjectives, adverbs, etc.) go along the best with a certain noun. Even though, noun classes are ordered hierarchically, in the parsing algorithm they are stored in a matrix that represents a weighted graph. There is a one-to-one value that relates each of the classes to each other, specifying if the fit is perfect, good, not so good or terrible. This threshold for terrible fits is used by the parser (see section 4.4.2.3.2).

For verbs, apart from the patterns, the following information is needed:

- All its conjugations (for each person and number) in these moods and tenses:
  - Present
  - Pretèrit perfet (similar to present perfect)
  - Passat perifràstic (past tense)
  - Present de subjuntiu (subjunctive)
  - Imperatiu (imperative)
  - Infinitiu (infinitive)
  - Gerundi (gerund)
  - Participi (participle)

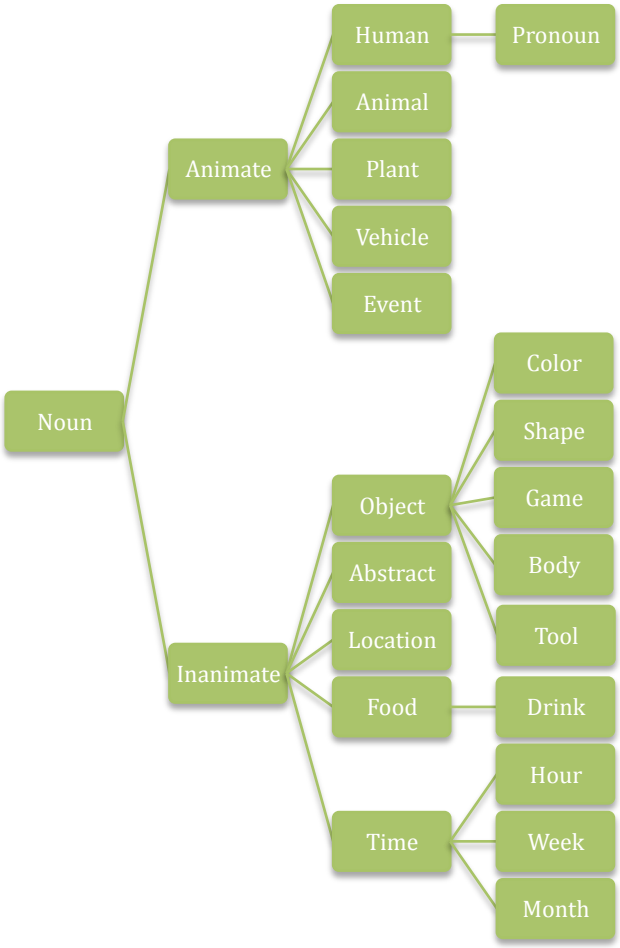



**Figure 8: Hierarchy of noun classes in the system. A noun can have several classes.**

For adjectives:

- All the forms of the adjective (masculine singular, feminine singular, masculine plural and feminine plural)
- All the adjectives classes, which help determine the nouns that it better modifier, from among the following:
    - All (non-specific)
    - Human
    - Animate
    - Object
    - Colour
    - Food
    - Ordinal (if the adjective is of the type ordinal numeral)

For adverbs:

- Type of adverb: manner, locative or time (only these types were present in CACE, though other classes may be constructed).

For modifiers[39]:

- All the forms of the modifier if it inflects (masculine, feminine, masculine plural and feminine plural)
- Type of modifier: quantifier, possessive, etc.
- Scope of the modifier: word or sentence modifier.

For question particles:

- Slots (1 or 2) that it fits. Most question particles can only fit a thematic role. However, there are some, like “què” (*what*) that can fit two (i.e. “Què vols?” (*What do you want?*); “Què ha caigut?” (*What fell?*); in the first example, “what” is the Theme and, in the second, it is the Subject). That is why up to two slots can be specified.

In order to increase the vocabulary of the system, we just need to add the information (or features) that we have seen above for the desired word (All the annotated words in the system can be found in digital appendix C). Therefore, even

[39] We have a special category for some Modifiers that, although most of them are adjectives, they are treated differently by the parser and the generator. The set of modifiers includes: quantifiers (not numerals), possessives and sentence modifiers, like “no”, “if”, “because”, etc.

though, the initial set of vocabulary, specially verbs, is somewhat reduced, as the system has been designed and built in order for it to be easily scalable (in terms of vocabulary available), it is not an issue and it can already represent the full potential and functionality of the system.

## 4.2. Input system

Now that we have seen the main decisions taken regarding the initial vocabulary for the system, we can continue to the first element that constitutes the *compansion* system, the input system. First of all, we would like to stress again that this input module has got the only purpose of feeding the parser and is not, by any means, to be considered the final input system, whose design and features are going to be described in section 4.6. This means that the current input system is not adapted in any way to the needs of AAC users. As it was already mentioned, the input does not use pictograms, but the words they refer instead.

When entering the system, for security reasons and to be able to easily gather the results for evaluating the *compansion* system, the user interface asks for identification access (Figure 9).

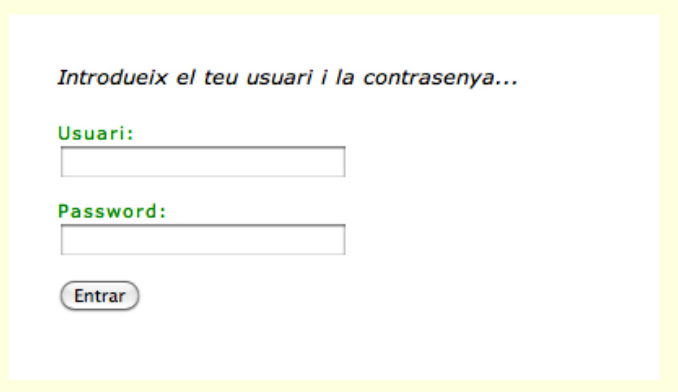



**Figure 9: Screenshot of the user interface: user login.**

On the next screen, the interface shows all the words and modifiers available and grouped by categories. Even though, the interface is not ready for AAC users, we still wanted it to resemble a little bit like an AAC system, so we decided to distribute words using common categories in AAC. Also, we used a similar colour notation that the one that is usually taken in AAC (see section 2.2), yellow for nouns that can be subjects, green for verbs, blue for adjectives, etc. (Figure 10).

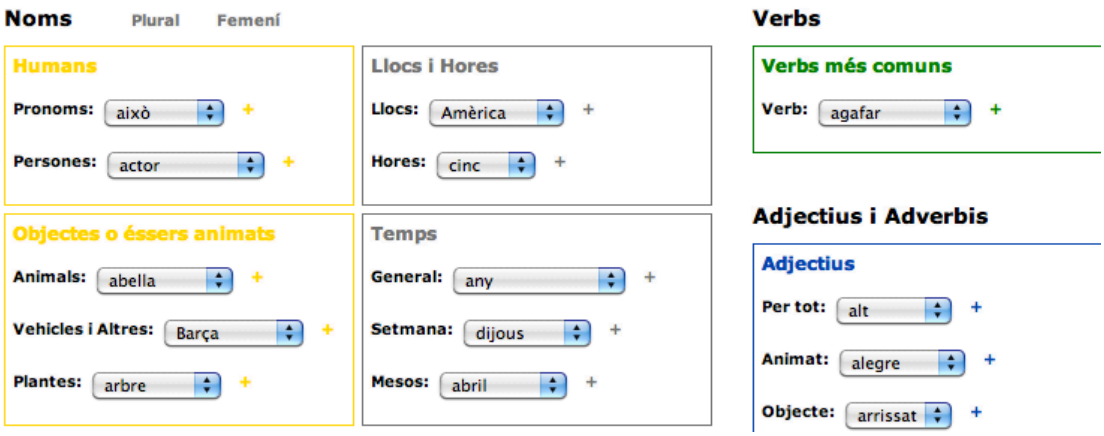



Figure 10: Screenshot of the user interface: Colour notation similar to AAC boards

In order to send an input that the *companion* system will immediately generate, these steps need to be followed:

1. Think of the sentence that you want to generate.
2. Choose the content words for the sentence from the lists.
    a. To add a word to the input, you need to select the word from the adequate list and the press the "+" button next to the list (Figure 11).


Figure 11: Screenshot of the user interface: Adding a word to the input

    b. Apply feminine, plural or coordination modifiers to a word, just after adding it to the input with buttons "Femení", "Plural" and/or "i" (Figure 12).


Figure 12: Screenshot of the interface: Adding feminine, plural or coordination modifiers to a word

    c. Added elements appear at the bottom of the screen and can be deleted individually by pressing the "-" button (Figure 13).


Figure 13: Screenshot of the interface: Selected elements

3. Choose the desired verb tense from the following options (or leave the default value) (Figure 14):

a. Defecte (*Default*): With this option, the generator will decide the best tense for the sentence depending on the type of sentence chosen, the verbs selected and the time expressions inputted, if any.
b. Present: This forces the main verb in the sentence to be in present tense.
c. Passat (*Past*): This forces the main verb to be in past tense.
d. Passat inmediat (*Immediate past*): The main verb will be in a tense similar to the present perfect tense.
e. Futur (*Future*): The main verb will be in the future tense.

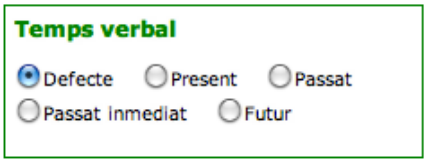



**Figure 14: Screenshot of the interface: Verb tense modifiers**

4. Choose the desired type of sentence from the following options (Figure 15):
   a. Enunciativa (*Declarative*).
   b. Desig (*Wish*): For sentence like "I want…", the system will add automatically "Vull" (*I want*) at the beginning of the sentence. When using this modifier, sentences with three verbs can be built. E.g. "Vull anar a comprar" (*I want to go shopping*).
   c. Demanar permís (*Ask permission*): For sentences like "Can I…", the system will add automatically "Puc" (*I want*) at the beginning of the sentence. When using this modifier, sentences with three verbs can also be built. E.g. "Puc anar a comprar?" (*Can I go shopping?*).
   d. Ordre (*Order*): If selected, sentences will use the imperative tense.
   e. Pregunta (*Question*): If a question particle has already been selected, the sentence is already a question and the type of sentence selected becomes overwritten.
   f. Resposta (*Answer*): It is mainly for sentences without a verb ("verbless"). If this option is selected, the system will give priority to patterns without a verb to produce sentences like "An apple, please".
   g. Condicional (*Conditional*): The generator will add the "si" (*if*) particle at the beginning of the sentence.
   h. Exclamació (*Exclamatory*).
   i. Negativa (*Negative*): Can be selected along with any other type of sentence. Transforms the sentence into its negative form.

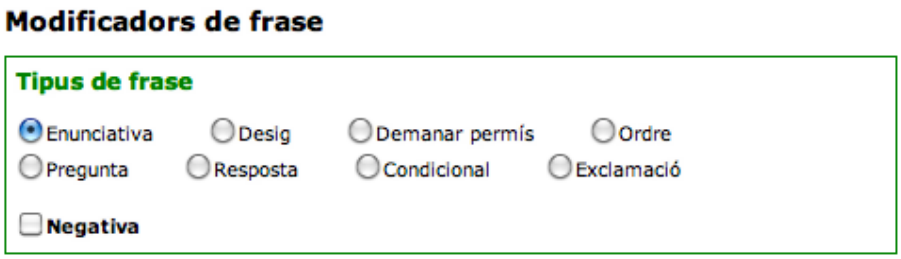



**Figure 15: Screenshot of the interface: Types of sentence**

5. Send the sentence by pressing the button "Generar" (*Generate*).
6. After sending the input, the parser will come into play.

Keep in mind that, apart from the previous steps, the system expects the user to try to build a sentence that makes sense and to take into account the restrictions in the sentence structure that we have detailed at the beginning of the Methodology section (section 4).

## 4.3. Design pattern and programming languages used

Before beginning the explanation of the parser, we would like to briefly explain the general design pattern that we have taken to build the whole *compansion* system. The application is built using the Model-View-Controller design pattern, where models access the database, views are the interface and controllers contain the main algorithms and feed the views with the necessary information. There are also libraries, which contain classes of custom-made objects that, in turn, can contain core functions of the system. In our case, libraries do contain important functions of the parser and generator.

Regarding the programming language, the application uses CodeIgniter[40] as a framework and has been coded in PHP. We have also used SQL for the definition of the database that holds all the annotated data, HTML for the structure of the user interface, CSS for the style of the interface and JavaScript for the dynamic behavior of the word selection process.

We decided to build it using web-based technologies because, this way, any device with Internet connection can access the system.

All the code developed by us during the project can be found, separated in folders, in the digital appendix D. In the code, there are comments that can help understand what is the

[40] CodeIgniter Framework – www.codeigniter.com [April 15th, 2012]

function of each part of code. The full code of the application within the framework can be found in digital appendix E.

## 4.4. Parser

The aim of our system is to transform telegraphic language that will eventually result from selecting pictograms into well-constructed natural language sentences. This is known as *cogeneration* [Copestake, 1997]. The job of a cogeneration system is to order text units from an input, add inflections and insert extra words (both function and content words). In fact, our system would not exactly do cogeneration, as it will not add content words, except in some special cases (i.e. if a user has selected a transitive verb and he has not chosen anything that can act as the theme (direct object), the system will insert a generic expression, such as "something" (see Figure 16), to fill in the gap, or if a user has not stated a subject, the system will use a default one depending on the verb and on the type of sentence). Apart from these special situations, the application will mostly do what is known as *compansion* [Pennington & McCoy, 1998] (compression-expansion). The main problem of both cogeneration and *compansion* is that frequent unexpected answers from the system get the user nervous and frustrated. Therefore, the system needs to have a high-rate of success in order to be really useful.

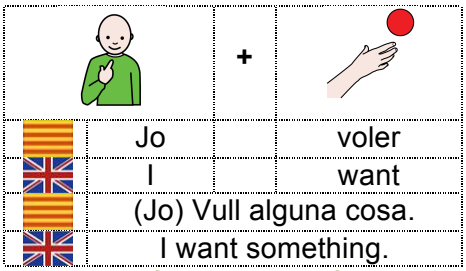



**Figure 16: Example of the insertion of a generic expression.**

*Compansion*, unlike cogeneration, only expands uninflected content words (compressed, synthetic or telegraphic utterances) into syntactically and semantically well-formed sentences.

Usually, in cogeneration systems, the user selects a certain sentence template (question, request, order, etc.) and then he has to fill the different slots that the interface provides in order to build the whole sentence. In our system the templates will be selected dynamically on-the-fly depending on the previous words selected by the user (mainly depending on the verbs selected, but also on sentence modifiers or question particles). Also, the templates or patterns will be invisible by the end user

and, if a certain verb selected can have several patterns associated, the system, in this case the parser, will select the best one according to the rest of the words selected.

Following the approach of Pennington and McCoy (1998) on *compansion*, the core of our *compansion* system is a semantic parser that interprets input based on the use of case frames or slots. These slots are conceptual structures that represent the meaning of a sentence by describing the semantic cases or roles that each of the content words has in relationship with the others. When running, the semantic parser designates the verb as the main component of the expression: all other words in the input are used to fill semantic roles with respect to the main verb that is chosen and the patterns annotated for that verb.

A verb pattern is defined by the following features:

- **Idverb**: Id of the verb that is going to be the centrepiece of the pattern.
- **Pronominal**: It indicates if the verb is pronominal or not. Pronominal verbs have the property that the receiver is always the same as the subject. In Catalan, these verbs have a pronoun that agrees with the subject and that always goes attached to the verb, either before or after the verb (depending on the verb form), e.g.:
  "Amagar-se" (*to hide oneself*), e.g.: "M'amago a l'armari" (*I hide in the closet*).
- **Pseudoimpersonal**: It indicates if the verb is of the type of "pseudo-impersonal" verbs. An example of it would be the verb "agradar":
  "M'agraden les pomes" (*I like apples*).
  Here, in Catalan, the subject is "pomes" (*apples*) and it usually goes after the verb, although:
  "Les pomes m'agraden" would also be correct.
- **Tipusdefrase**: It indicates the default type of sentence for that pattern. Here the types of sentence are the one that we saw that are selectable as modifiers in the input system.
- **Defaultense**: It indicates the default tense for the pattern. The values are the same as the ones that can be selected in the input system. Besides, there is an extra value named "verbless" that indicates that the pattern does not have a verb as a root. This type of patterns will allow the system to produce sentences like "una casa" (*a house*), "la meva mare" (*my mother*), "content" (*happy*), "més" (*more*), etc. that are usually replies and do not have any verb instantiated.

- **Subj**: It indicates the type of subject that better fits this pattern, in other words, if the subject should be “human”, “animate”, “object”, any “noun” or even a “verb” phrase. If this pattern does not have subject, the value will be ‘0’.
- **Subjdef**: It indicates the default person value for the subject (first, second or third person singular), in case the user omits it in the sentence. In most cases, in the context of AAC, users are talking about themselves, therefore the default person value for the subject is the first person singular.
- **Theme**: It indicates if the semantic role of THEME[41] is mandatory (1), optional (opt) or whether it cannot appear in the pattern (0). When the verb is copulative, this slot becomes the ATTRIBUTE slot. For example:
  “Vull una poma” (): Here the theme is mandatory.
  “Bec coca-cola” (*I drink coca-cola*): Here the theme is optional.
  “Vaig a l’escola” (*I go to school*): Here there cannot be theme. “A l’escola” (*to school*) would be location destiny (LocTo) slot.
  “Ell és alt” (*He is tall*): Here the theme slot would become an attribute slot.
- **Themetipus**: It indicates the type of theme that better fits this pattern. It can be a noun from the set of classes of nouns available in the system (“human”, “object”, etc.) (Figure 8, page 27), an adjective (when the slot becomes an attribute slot), a verb phrase, etc.
- **Themedef**: If the theme is mandatory, it indicates the default value for it.
- **Themeprep**: If not empty, it indicates the preposition that precedes the Theme slot. This feature is defined for the generator.
- **Receiver**: It indicates if the semantic role of RECEIVER is mandatory (1), optional (opt) or cannot appear in the pattern (0). The receiver does not have the feature ReceiverType because the system considers that the best receiver is always an animate noun.
- **Receiverdef**: If the Receiver was mandatory, it indicates the default value for the Receiver.
- **Receiverprep**: If not empty, it indicates the preposition that precedes the Receiver slot. This feature is designed for the generator.

[41] The names for all the slots are orientatory and do not exactly correspond to the semantic concepts. They are named like that to facilitate the creation of patterns. What is important to the parser are the values of the features, basically the importance of the slot (mandatory, optional, forbidden) and what semantic class can fit the slot (e.g. a noun of the “human” class). Here the theme slot corresponds to the semantic role of theme, but also to the role of patient and of attribute, in copulative verbs. On the other hand, the generator considers this theme slot, mainly as the syntactic direct object.

- **Benef**: It indicates if the semantic role of BENEFICIARY is mandatory (1), optional (opt) or cannot appear in the pattern (0).
- **Beneftipus**: It indicates the type of beneficiary that better fits this pattern from the set of classes of nouns available in the system ("human", "animate", etc.).
- **Benefdef**: If the beneficiary was mandatory, it indicates the default value for the beneficiary.
- **Benefprep**: If not empty, it indicates the preposition that precedes the beneficiary slot. This feature is defined for the generator.
- **Acomp**: It indicates if the semantic role of COMPANY (companion) is optional (opt) or cannot appear in the pattern (0). The company slot does not have the feature CompanyType because the system considers that the best companion is always an animate or human noun.
- **Acompdef**: If the Company was mandatory, it indicates the default value for the Company.
- **Acompprep**: If not empty, it indicates the preposition that precedes the Company slot. This feature is defined for the generator.
- **Tool (Instrument)**: It indicates if the semantic role of INSTRUMENT is optional (opt) or cannot appear in the pattern (0). The Tool slot does not have the feature ToolType because the system considers that the best instrument is always a tool, a vehicle or an object noun.
- **Tooldef**: If the Tool was mandatory, it indicates the default value for the Tool.
- **Toolprep**: If not empty, it indicates the preposition that precedes the Tool slot. This feature is defined for the generator.
- **Manera**: It indicates if the semantic role of MANNER is optional (opt) or cannot appear in the pattern (0).
- **Maneradef**: If the Manner slot was mandatory, it indicates the default value for it.
- **Maneratipus**: Usually, the manner slot is better filled with an adverb of manner or certain adjectives, when they act as adverbs. E.g.:

  "Fes-ho ràpid" (*Do it fast*).

  However, sometimes, we considered that this slot can also be filled by quantifiers. Maybe then it should be considered another type of slot, but in order to have a small number of slots, in order to simplify the parser, we joined them. E.g.:

  "M'agraden molt les pomes" (*I like apples very much*).

  This feature indicates if a quantifier would also be a good fit for the slot.

- **Locto (Destiny)**: It indicates if the semantic role of LocTo is mandatory (1), optional (opt) or cannot appear in the pattern (0).
- **Loctotipus**: If empty, the system understands that the best fit is a noun of the location class. Else, it indicates the best fit for the LocTo slot in that pattern (another class of noun, a verb phrase, etc.).
- **Loctodef**: If the LocTo slot was mandatory, it indicates the default value for the LocTo slot.
- **Loctoprep**: If not empty, it indicates the preposition that precedes the LocTo slot. This feature is defined for the generator.
- **Locfrom (Source)**: It indicates if the semantic role of LocFrom is mandatory (1), optional (opt) or cannot appear in the pattern (0).
- **Locfromtipus**: If empty, the system understands that the best fit is a noun of the location class. Else, it indicates the best fit for the LocFrom slot in that pattern (another class of noun, a verb phrase, etc.).
- **Locfromdef**: If the LocFrom slot was mandatory, it indicates the default value for the LocFrom slot.
- **Locfromprep**: If not empty, it indicates the preposition that precedes the LocFrom slot. This feature is defined for the generator.
- **Locat**: It indicates if the semantic role of LocAt is mandatory (1), optional (opt) or cannot appear in the pattern (0). The LocAt slot does not have the feature LocAtType because the system considers that the best LocAt is always a noun of the class of location.
- **Locatdef**: If the LocAt slot was mandatory, it indicates the default value for the LocAt slot.
- **Locatprep**: If not empty, it indicates the preposition that precedes the LocAt slot. This feature is defined for the generator.
- **Time**: It indicates if the semantic role of Time is mandatory (1), optional (opt) or cannot appear in the pattern (0). Usually, in most patterns time expressions are optional.
- **Expressio**: If there is an expression written, it is a default expression that will always appear in that pattern. For example, the patterns for the verb “ajudar” (*to help*), have the expression “si us plau” (*please*) as default.
- **Subverb**: This feature indicates if the type of any of the slots of the pattern is another verb. In other words, it says if the pattern accepts another verb as a complement or not.

| Donar | |
|---|---|
| Pronominal | 0 |
| Pseudoimpersonal | 0 |
| Tipusfrase | ordre |
| Defaultense | imperatiu |
| Theme | 1 |
| Themetipus | human |
| Themedef | una cosa |
| Themeprep | |
| Receiver | 1 |
| Receiverdef | mi |
| Receiverprep | a |
| Benef | 0 |
| Benefdef | |
| Benefprep | |
| Acomp | 0 |
| Acompdef | |
| Acompprep | |
| Tool | 0 |
| Tooldef | |
| Toolprep | |
| Manera | opt |
| Maneradef | |
| Maneratipus | adv |
| Locto | 0 |
| Loctotipus | |
| Loctodef | |
| Loctoprep | |
| Locfrom | 0 |
| Locfromtipus | |
| Locfromdef | |
| Locfromprep | |
| Locat | opt |
| Locatdef | |
| Locatprep | a |

| Donar | |
|---|---|
| Time | opt |
| Expressio | si us plau |
| Subverb | 0 |

Table 2: Example of a pattern of the verb “To give”.

In Table 2, we can see an example pattern of the verb “Donar” (*To give*). In this case, we can see that the default subject is not the first person, but the second person. We think that it is more reasonable in the context of use of the AAC device and in normal life contexts for the subject or agent of the verb “to give” to be the second person singular “you” and for the verb to be in an imperative form. Therefore, if the user wanted to use the first person singular, he/she should explicitly use the pronoun “jo” (*I*) in the input.

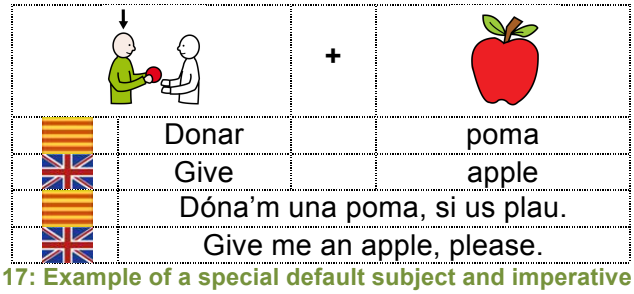



Figure 17: Example of a special default subject and imperative verb form.

When defining a pattern, if we needed a type of semantic role that is not in the features list, we could always use the features from an unused slot, which does not have special syntactic properties (i.e. the subject, which can be reduced and which usually appears before the verb, or the receiver, that can be pronominalised), that would act as the desired slot instead.

This is possible because in the system, we have prioritised functionality and performance over linguistic fidelity in the methodology.

### 4.4.1. General characteristics

Having seen the basics of the system and the structure of the verb patterns (all the annotated patterns can be found in appendix B), we will now start defining the dependency-parsing algorithm. As we already mentioned at the beginning of section 4 (page 22), the parser uses a controlled or constraint (in the less technical sense of the term) grammar dependency parsing method. Controlled grammar because the *compansion* system does not accept all the existent constructions in Catalan, but a restricted set of it. The approach that we use is a grammar-driven eliminative approach (Karlsson, 1990), but, instead of just using syntactic information for the constraints, we use a mix of syntactic language-dependent information, plus

semantic features, annotated in the information for each word in the database, meaning that our system combines different levels of analysis, similar to the notions that can be found in Extensible Dependency Grammar (Debusmann et al., 2004).

The system also does labelled dependency parsing, as the dependencies from head to dependent are tagged using the semantic roles on the first level of the tree and then tags such as “NC” for noun complement, “ADJ” for adjectives that act as complements, “ADV” for adverbs and “MOD” for other modifiers on the rest of the levels.

As we can see in Figure 18, the idea is to limit to two levels the dependency tree, to ensure simple sentence structures and to limit possible errors that would lead to wrongly expanded sentences. The two-level constraint does not take into account extra levels that can result from adding prepositions or articles and other determiners to nouns that appear on the second level during the generation. It does not take into account either a third extra level that can appear in sentences where a second verb acts as the nucleus of a thematic role and thus depends on the root verb.

Also, the method to label dependencies is deterministic, similar to the one that Covington (2001) explains. In general, our system starts searching the possible dependencies of the root. Once it has all of them, it proceeds to the second level of the tree looking for possible dependencies, constrained by the grammatical categories of the heads of the first level and so on with the rest of the levels.

Finally, in case a word can depend on several heads or can fit several slots, in order to disambiguate, semantic features [Karberis & Kouroupetroglou, 2002] of the head and the words that can modify it (in the first case) or of the slot and the words that can fit it (in the second case) are used (we will see this mechanism in more detail in section 4.4.2). After applying these features, if there are still ambiguities left, only then language-dependent word-order syntactic information is used. Also, to disambiguate different correctly parsed sentences that come from different possible patterns for the same verb, a weighting method is used (section 4.4.2.4).

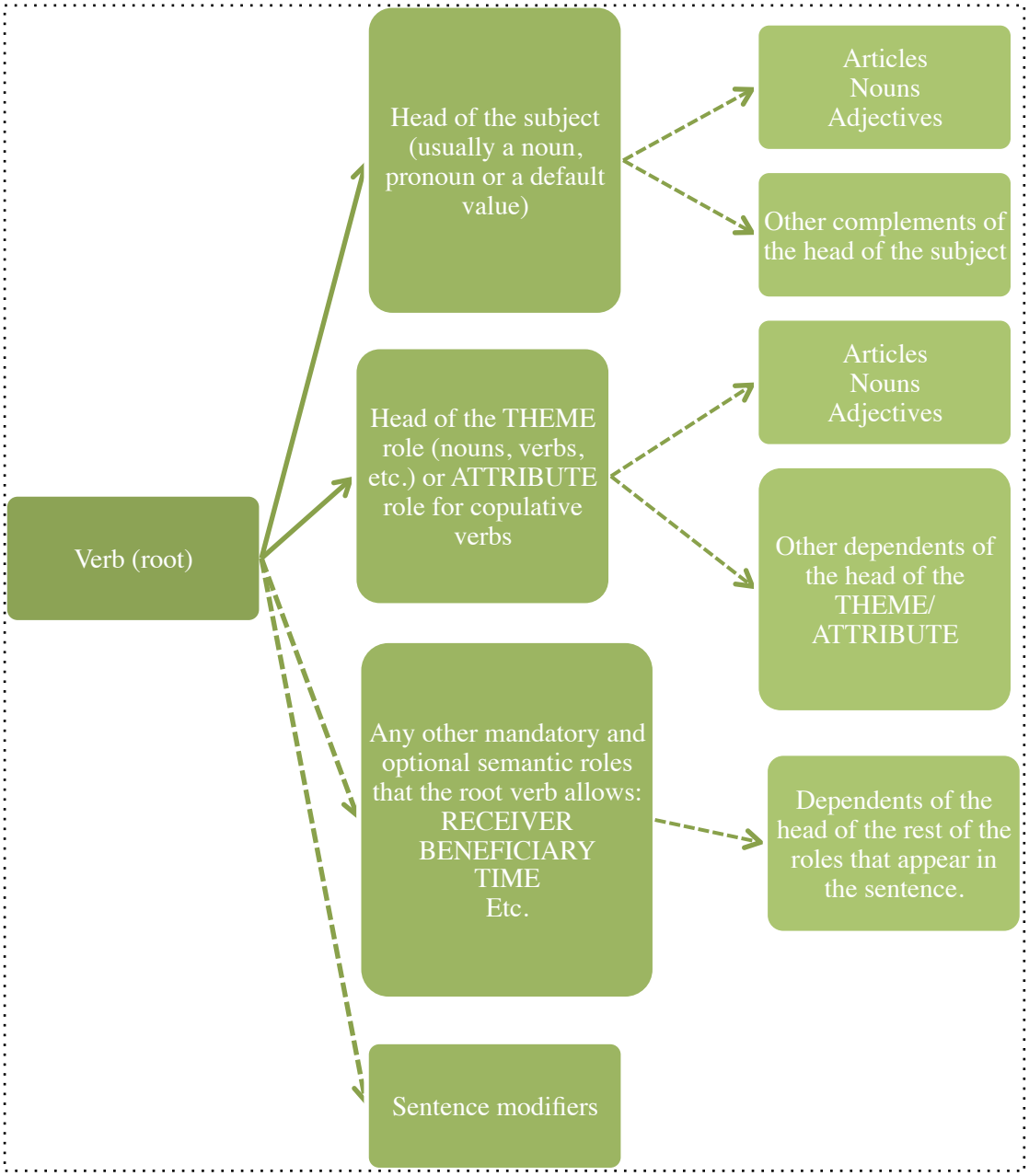



**Figure 18: Example of a 2-level tree that represents a simple sentence structure. Labels have not been added to ensure legibility. Dotted dependencies can appear multiple times.**

### 4.4.2. Parsing algorithm: Steps and rules encoded in the parser

After seeing the main characteristic of our dependency parsing approach, we now proceed to present the parsing algorithm and its rules. Here are the rules that the parser applies in order of appearance in the algorithm:

#### 4.4.2.1. Initialisation

1. Retrieves all the information on the words inputted by the user and initialises the data structure for each of the words, taking into account the possible modifiers applied by the user.

2. If a word has a coordination modifier applied, it checks if the following word is of the same type to see if both can be coordinated. If so, the second word becomes transparent to the parser, as both words will fit the same slot. The second word will not appear until the generation steps.

#### 4.4.2.2. Retrieval of the verb patterns

1. If there are no verbs in the input, the parser retrieves the "verbless" patterns.
    a. If the type of sentence is not an answer, it also retrieves the patterns for the verbs "ser" i "estar" (*to be*). This is to be able to make sentences like "Estic content" (*I am happy*) by only inputting the words or pictograms for "jo" (*I*) and "content" (*happy*), therefore omitting the verb "to be".
2. If there is one verb in the input, the parser retrieves all the patterns available for that verb, except for the patterns that needed a second verb. For example, if there was the verb "anar" (*to go*), the pattern for sentences like "Vaig a comprar" (*I go shopping*), which need a second verb, would not be retrieved. Only the patterns like "Vaig a casa" (*I go home*), where the head of the LocTo slot is a noun and not a verb, would be retrieved.
3. If there are two verbs, the system decides which is the main verb and which is the secondary verb, depending on which one can have a secondary verb as a complement. Then it retrieves the patterns for both verbs (for the main verb, only patterns that require a secondary verb and, for the secondary verb, only patterns that do not require a second verb) and fuses the patterns of the secondary verb in the slot that required a verb as the head of the slot in the patterns from the main verb. This is done M*S times, where M is the number of patterns from the main verb retrieved and S is the number of patterns from the secondary verb retrieved. The slots of the newly formed pattern are named as seen in Figure 19.

#### 4.4.2.3. For each pattern

##### 4.4.2.3.1. Beginning

1. The parser separates all words in the input by types: Nouns, Adjectives, Adverbs, Expressions, Modifiers (this category includes numbers) and Question particles.
2. Put the expressions in the expressions slot, if any.
3. Put the question particle in its slot, if any. Only one question particle per sentence is allowed.

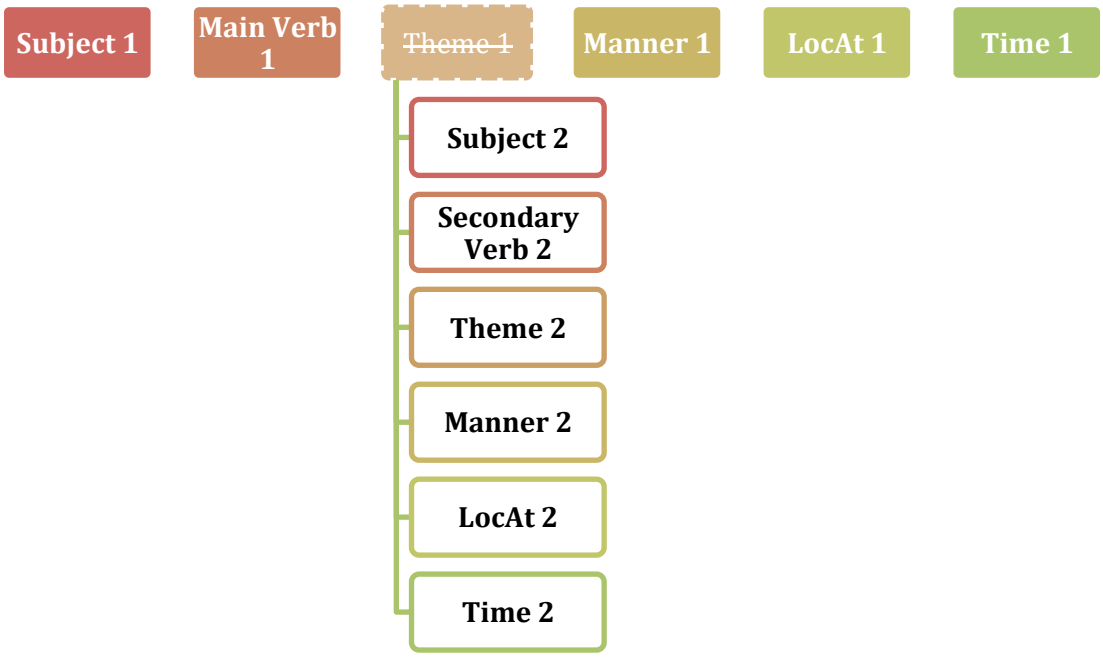



**Figure 19: Example of fusion of patterns for inputs with two verbs. The slot that needed a verb disappears and is replaced by the slots of the pattern of the Secondary verb.**

4. If the main verb is "pseudo-impersonal" or there is a question particle, the preferred order of the subject in the sentence is inverted. The more common position of the subject in sentences with "pseudo-impersonal" verbs or with question particles is after the verb, as opposed to the most common position of the subject that usually goes before the verb. Thus, the criteria for disambiguating will change in these cases.

#### 4.4.2.3.2. Solve Nouns

The parser starts to try to fill the slots of the pattern with nouns because they are the most common heads for slots. Also, they are the type of words that can have more complements, so once they are placed it will be easier to assign them the complements.

The process to place the nouns in the slots works as follows:

1. For each noun (N), see if it can fit each of the slots of the pattern.
    a. N can only fit slots that require a noun as the head of the slot[42].
        i. If it can fit the slot, depending on the class of noun that the slot requires and depending on the class (or classes) of noun that N is, the provisional fit is given a score (see Figure 20).
    b. Also see if N can act as a complement for another noun that is fitting temporarily or definitely a slot.

[42] A perfect fit occurs when the word that fits the slot has the exact same class as the class that better fits a slot, which is specified for each slot in the verb pattern.

i. A pronoun cannot have any complements.

c. After this process, if N can only fit a slot (be it as a head or as a noun complement), then N is assigned that slot and that slot is blocked.

**The scoring works as follows:**

- If the slot is mandatory, its initial score is higher (except for Subject slots that have the same initial score as optional slots).
- The worse the fit is, in other words, the further the class that the slot prefers is to the class of the word that fits it, the more points are subtracted to the initial score.
- If it is a Subject slot and the word that fills it is not a terrible fit[43] and it was inputted sequentially before the verb, extra points are given to the fit in order to level it to the importance of mandatory slots.

**Figure 20: Rules that define how the parser scores the fit of a word with a certain slot.**

2. The parser starts the disambiguation process for all the slots that can have several fits and that are not blocked yet (slots that act as a noun complement do not enter this process).
    a. The parser sorts all slots by the score of the noun that better fits each slot. The slot with the best score is the one that is treated first.
        i. If two nouns fit the slot equally well, the one that was input first is taken over the other.
        ii. If two slots have the same score, the one that appears first in the pattern is taken over the other (slots in the pattern are found in the most common order for the constituents in Catalan). Except for:
            1. If there is a secondary verb, slots of the secondary verb have priority over slots from the main verb.
            2. Receiver slots always have priority over Subject slots (the parser normally reaches this point of disambiguation when a personal pronoun is found in the input and the pattern has both Subject and a mandatory Receiver slot. If there is a single personal

[43] The parser uses a threshold for terrible fits. Terrible fits are encoded in the values of the graph that represents how semantic classes fare with each other (see the beginning of page 49 for more details and digital appendix D, file “Mymatching.php” in folder “libraries”, to see the matrix that represents the graph).

pronoun, we decided to prioritise it filling the Receiver, as speakers can omit the Subject, but not the Receiver if it is mandatory, and we expect that users will also reflect this behaviour in the input).

   b. To treat a slot, the noun (N) that better fits it is assigned that slot and that slot is blocked. Also N is erased from all the other slots that it could potentially fit.
   c. Steps "a" and "b" above are repeated for the remaining nouns and slots until there are not any more slots to disambiguate.

3. Once all slots are disambiguated, the parser checks if a noun would be better as a noun complement than as a fit for an optional slot.
   a. Nouns that fit mandatory slots are not checked.
   b. Nouns that perfectly fit[44] an optional slot are not checked either, unless both appeared before the verb.
   c. Slots that already have a noun acting as a noun complement are not checked.
   d. The parser considers that a noun (N1) would be better as the complement of another noun (N2) if criteria "a", "b" and "c" are not met and if N2 directly precedes N1 in the input sequence.
   e. If the previous conditions are met, then N1 will act as a complement of N2 and not as the head of the optional slot that it filled.

#### 4.4.2.3.3. Solve Adverbs

We decided to solve adverbs before adjectives, because both adjectives (when they behave as adverbs; see end of page 36) and adverbs can fit manner slots and we wanted to give preference to adverbs filling manner slots over adjectives. That is only relevant, if any manner adverb is present in the input. Else, the decision of solving adverbs before adjectives would be irrelevant, as they do not collide in any other occasion.

The process to place the adverbs goes as follows:

1. For each Adverb (ADV):
   a. If it is an adverb of time, it is added to the slot of time expressions, which is a list of time expressions[45]. The parser accepts several time expressions appearing in the same sentence, even though this may not be entirely adequate from a linguistic point of view.

[44] See footnote on page 65.

[45] This slot is an exception, as in the same slot more than one time expression can fit.

b. Else, see if ADV can fit each of the slots of the pattern:
    i. ADV can only fit slots that require an adverb as the head of the slot.
    ii. Locative adverbs cannot fit in slots that require an adverb of manner and vice versa.
    iii. The parser also checks if locative adverbs can go as a complement of a noun that is the head of a Location slot. E.g.: "És a sota la taula" (*It is under the table*).
    iv. After this process, if ADV can only fit a slot (be it as a head or as a noun complement), then ADV is assigned that slot and the slot is blocked.

2. The parser starts the disambiguation process that works exactly in the same way as the one used for nouns.
3. Finally, once all slots are disambiguated, the system checks if there is a locative adverb (ADV) that better goes as a noun complement of the head of a Location slot (N), instead of as a fit for an optional slot.
    a. Adverbs that fit mandatory slots are not checked.
    b. Slots that already have an adverb acting as a noun complement are not checked either.
    c. If there are several adverbs that can act as a noun complement for N, the system selects the adverb that is closer in distance (in the sequence of inputted words) to N. If there is a tie between two adverbs, adverbs that appear at the left side of N, which is the natural position for a location adverb in Catalan, have priority.
    d. If the previous conditions are met, then ADV will act as a complement of N and not as the head of the optional slot that it filled.

#### 4.4.2.3.4. Solve Adjectives

We decided to solve adjectives before modifiers because adjectives accept some type of modifiers, like quantifiers, and, thus, we need to place adjectives first.

The process to place the adjectives goes as follows:

1. For each Adjective (ADJ), see if it can fit each of the slots (S) of the pattern:
    a. ADJ can only fit slots that require an adjective as the head of the slot. However, the system also accepts some adjectives as the head of Manner slots, for example:
    "Va fer-ho molt ràpid" (*He did it very fast*).
    b. The parser also checks if ADJ can function as a complement of a noun (N) that is the head of slot S.

i. The parser considers that any adjective can go as a noun complement of any noun[46]. Nevertheless, the parser gives a score to the fit that has two features into account:
    1. The distance between ADJ and N in the input. If ADJ appears just after N in the input, it has an additional point.
    2. How well ADJ fits N, taking into account the class of N and the class of ADJ.
ii. The highest the fit, the highest is the punctuation.

c. If after this process ADJ can only fit a slot (be it as a head or as a noun complement), ADJ is assigned that slot and the slot is blocked.

2. The parser starts the disambiguation process that works exactly in the same way as for nouns and adverbs (see section 4.4.2.3.2, page 44).
3. Finally, once all slots are disambiguated, the system checks *for each* adjective (ADJ) that is not already fitting a mandatory slot, which noun (N) they fit better as a noun complement.
    a. To do so, the scores explained on step (1bi) above are taken into account.
    b. The fit with the highest score is taken, always checking that N does not already have another adjective as complement (remember that our parser only accepts one adjective complement per noun).
    c. If ADJ is already fitting an optional slot, the system checks if it better fits N or the optional slot.
    d. Finally, if being a noun complement is the best fit for ADJ, ADJ will be parsed as a complement of N.

#### 4.4.2.3.5. Solve Modifiers

Reached this point, modifiers are the only type of word remaining to be placed (along with numerals that the parser has already grouped with modifiers).

The process to place the modifiers goes as follows:

1. For each Modifier (MOD):
    a. If it is a phrase modifier, and not a word modifier, and the pattern is not “verbless”, MOD is assigned to the main verb. If the pattern is “verbless” the sentence modifier cannot be assigned to the verb, as there is not one, and the only option for it is to be placed in a slot that

[46] Although adjectives are usually not complements but modifiers, for practical purposes, the parser deals with adjectives in a manner similar to noun complements.

requires a modifier as the head. This can happen in replies where the answer is a single modifier, like “No!” (*No!*).

b. Else, see if MOD can fit each of the slots of the pattern:
    i. MOD can only fit slots that require a modifier as the head of the slot, which is quite rare. For example, in a “verbless” pattern that only has one slot. This would be to produce sentence with a single word that would be the modifier (i.e. “Molt” – *A lot*)
    ii. However, if MOD is a quantifier, it can also fill slots that require a quantifier as the head.
    iii. Nevertheless, what is mainly checked at this step is if MOD can act as complement of the head (H) of the slots, be it an adverb, a noun (not a pronoun), an adjective or even another modifier.
        1. However, if MOD is a possessive or a numeral, it can only act as a complement of a noun.
        2. If MOD can indeed act as a complement, a score, on how good of a complement it is, is given based on the distance between MOD and H, the head of that slot.
    iv. After this process, if MOD can only fit a slot (be it as a head or as a noun complement), then MOD is assigned that slot and the slot is blocked.

2. The parser starts the disambiguation process that works exactly in the same way as for nouns, adverbs and adjectives (see section 4.4.2.3.2, page 44).
3. Finally, once all slots are disambiguated, *for each* modifier (MOD) that is not already fitting a mandatory slot, see which head (H) they fit better as a complement.
    a. To do so, the scores explained on step (1biii2) above are taken into account.
    b. The fit with the highest score is taken, always checking that if MOD is a quantifier, H does not already have already an adverb as complement (remember that our parser does not accept modifiers as complements if an adverb is already complementing H; Also remember that a single head can have several modifiers applied to it).
    c. Finally, if a head to complement was found for MOD, MOD will act as a complement of H.

#### 4.4.2.3.6. Total points for the pattern

1. Once all words are treated, the parser gives an overall score to the pattern. The scoring scheme works as follows:
    a. The parser starts with an arbitrary total score.
    b. For all slots filled (mandatory slots, optional slots and complement slots), the score for that filled slot is added to the total score. Remember that every time that a slot is filled it is given a score depending on how good the word filling the slot is for that particular slot.
    c. For all mandatory slots not filled, a severe penalisation is applied.
    d. For all optional slots not filled, a very light penalisation is applied.
    e. For all words in the input not used (it is a possibility that no slot was found for a certain word in the input) a severe penalty is applied.

### 4.4.2.4. Choose the best pattern

1. Once all patterns are treated, the parser chooses the pattern with the highest score.
2. If two patterns have got the same score, the system selects the first one. In the annotated data, patterns for a given verb appear intuitively in order, from the most common pattern in terms of usage in speech to the less common one.

Once the system has the selected pattern, it will apply the generation algorithm to it. An example of the result of the parsing algorithm could be the following. From the input:

(1) Ahir donar voler nena ós peluix vermell.

(*Yesterday give want girl bear teddy red*)

A recreation of the parsed output is on Figure 21 (dependency labels are in brackets).

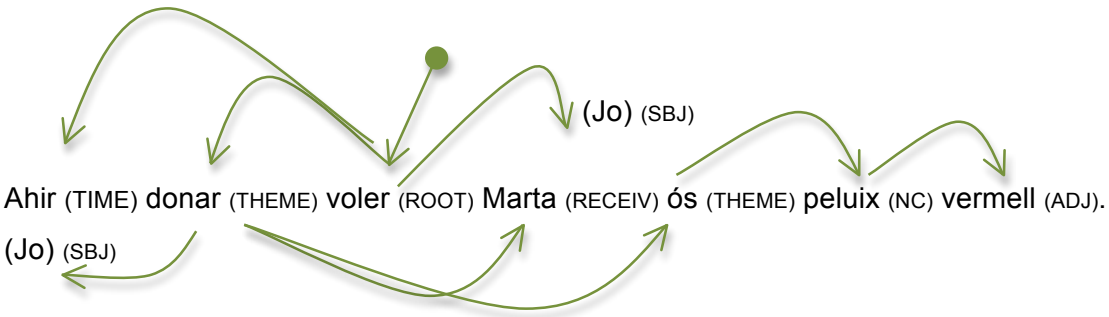



**Figure 21: Recreation of the parse tree of an example input sentence.**

Now that we have seen the behaviour of the parser, it is interesting to remark that the parser is nearly language independent, as it relies more on semantic features than in syntactic information. The system only takes into account language dependent word order, when it needs to disambiguate or for nouns that act as noun complements, which the system restricts to nouns that immediately follow other nouns (as we have seen in section 4.4.2.3.2). This means that adapting the parser to other languages, like Spanish, would just involve minor changes to the code (apart from the tagging of all the vocabulary).

## 4.5. Generator

Even though at this point the parsing process has already finished, in order to obtain the full natural language sentence, a generation algorithm is needed.

The generator in the system is built using independent modules, each with a specific task, that progressively expand the parse input step by step. As opposed to the parser, the generator uses mainly syntactic information to expand the parsed input into a natural language sentence in Catalan. Most of the modules are tailored to the controlled grammar of our *compansion* system. Nevertheless, they have beem designed in such a way so that they can be improved in the future and that they would be able to accept new structures.

In the next subsections, we will explain the aim of each of these modules, how do they work and which grammatical rules they apply in the generation process.

### 4.5.1. Preliminar steps

Even though the parser already used information on some sentence types, in some cases, the sentence type was not used and might not be set correctly[47]. What this step does is to override the default sentence type in the following occasions:

1. If there is a question particle in one of the slots of the pattern, the system sets the type of sentence as “pregunta” (*question*).
2. Else, if the default tense of the main verb is the imperative and no other sentence modifier (either tense modifier or type of sentence modifier) has been specified, the system sets the type of sentence as “ordre” (*order*).

[47] Sentence types coming from the sentence modifiers of the input are already set correctly.

### 4.5.2. Slot ordering module

This module orders the components (slots) of the sentence according to the type of sentence (declarative, question, order, wish, etc.) that we are generating.

1. The module checks if the are adverbs of time that need to go at the beginning of the sentence. Some adverbs of time are usually placed at the beginning of the sentence and some at the end. For example, “avui” (*today*) usually goes at the beginning: “Avui aniré al cine” (*Today, I will go to the cinema*), but “tard” (*late*) always goes at the end: “Vindré tard” (*I will come late*).
2. If there are two verbs in the sentence, and the subject for the secondary verb was not defined by the parser, if its default value was "jo" *(I*), then the subject for the secondary verb will get the same value of the main verb, be it either its default value or the word that fits it. This decision was made so that building sentences like “La meva mare vol anar de vacances” (*My mother wants to go on holiday*), where the subject of the main verb and the secondary verb are the same, would be easier as the user will only need to enter the subject “My mother” once. In other words, the generator takes into account zero anaphora.
3. The system puts the rests of the slots in the generic order: Subject – Verb – Theme – Receiver – … – Location slots – Time – Expressions. It also fills the mandatory slots of the pattern that were not filled, if any, with their default values.
4. The system searches for the pronoun “ho” (*it*) at the Theme position. If it exists, the system puts it just after the verb if the sentence is an “order” and it is not negative, else it puts it just before the verb. Examples: “Fes-ho” (*Do it*); “No ho facis” (*Don’t do it*); “Ho sé” (*I know (it)*).
5. The system also looks for pronominal Receivers. If it finds any, it puts it just after the verb if the sentence is an “order”; else it puts it just before the verb. Here are some examples of how the generator orders pronominal Receivers: “Explica-li-ho” (*Explain it to him/her*); “Li ho dono” (*I give it to him/her*); “Em dones la mà” (*You give me your hand*).
6. If the sentence has the modifier “Desig” (*Wish*), it adds a slot with the verb “voler” (*to want*) at the beginning of the sentence. Also, if it has a modifier of “Permís” (*Permission*), it adds a slot with the verb “poder” (*can*) at the beginning of the sentence.
7. If it has the modifier “Condicional” (*Conditional*), the generator adds the particle “si” (*if*) at the beginning of the sentence.

8. If there is a question particle, the system moves it at the beginning of the sentence.
    a. If the question particle is “quant” (*how many*), the system swaps the position of the Theme and the Subject. For example, “Quants anys té la nena?” (*How old is the girl?*).
    b. Else, the generator puts the subject at the end of the sentence. E.g. “Amb qui va anar al cine la mare?” (*Whom did mom go to the cinema with?*).

#### 4.5.3. Word order module, prepositions module and agreement module

Inside each filled thematic role, this module will put the head and its dependents in the correct order, depending on the grammatical category of the head and the dependents (including quantifiers and possessives). It also includes coordinated words that were transparent to the parser, adds the necessary prepositions and makes sure that words agree in gender and in number (e.g. attributes with the subject, modifiers with their heads, etc.).

1. First, this module gets the gender of the number of the subjects in the sentence (0, 1 or 2 depending on the number of verbs in the sentence). If there are two subjects, one for the main verb and the other for the secondary verb, it also checks if the subjects are the same.
2. For each slot, it orders the words inside the slot depending on the type of head (H) and makes sure that the modifiers agree with H in the following way:
    a. If H is a Noun the relative order of the elements inside the slot is (articles will be added by another module later on):
        i. PREPOSITION: Mostly taken from the patterns. In some cases like time expressions or noun complements, it is directly encoded in the generator.
        ii. QUANTIFIER and/or POSSESSIVE: If H also has an adjective as a complement, the system checks if the quantifier would better go with the adjective (e.g. “L’home més alt” (*The tallest man*)). If so, it keeps the quantifier for later. Else, quantifiers are placed before H. Possessives are always placed before H. Both quantifiers and possessives agree with in gender and number.
        iii. LOCATIVE ADVERB: Are also placed before the noun. E.g. “A sota la taula” (*Under the table*).

iv. NUMERALS: Also before the noun. E.g. “Els meus cent euros” (*My hundred euros*). If there is a numeral, H becomes plural (except for number 1). Numbers 1 and 2 also agree in gender with H.
v. ORDINALS: Also before H. E.g. “El primer jugador” (*The first player*). Ordinals agree in gender and number with H.
vi. HEAD (+ Coordinated word): Then the system adds the head noun plus its coordinated word (if any).
vii. ADJECTIVE and/or NOUN acting as a noun complement: If H has both an adjective (ADJ) and a noun (N) acting as complements, the generator checks if ADJ better fits N over H. If so, it puts first, just after H, N and the ADJ. Else it puts ADJ first and then N. If there is just ADJ or N complementing H, and not both, the system puts them just after H. Also, if any of them had a coordinated word, the generator would also add it. In any case, if ADJ had a quantifier kept from step “ii”, the quantifier is added before ADJ. Adjectives agree in gender and number with H.

b. If H is an Adjective, the order is the following:
i. PREPOSITION: If there is any.
ii. QUANTIFIER: If any. They do not agree, either in gender or in number.
iii. HEAD (+ Coordinated word): Adjectives can only be the head of slots pertaining to copulative verbs or to “verbless” patterns (except if they fit slots of Manner). In both cases, they need to agree in gender and in number with the subject, if any. Nevertheless, if the user applied gender or number modifiers from the input, they have priority over the existent gender and number of the subject. However, if the adjective is fitting a slot of Manner, it also appears in its masculine form. E.g. “La Maria va acabar-ho molt ràpid” (*Mary finished it very fast*).

c. If H is a Modifier:
i. PREPOSITION: If there is any.
ii. HEAD: Which is a modifier.
iii. QUANTIFIER: If any.

d. If H is an Adverb:
i. PREPOSITION: If any.

ii. QUANTIFIER: If any.
iii. HEAD: Which is an adverb.

e. If H is a Question particle:
    i. HEAD: Without any preposition. This slot just has the question particle.

f. If H is any other type of word (i.e. a verb):
    i. PREPOSITION: If any.
    ii. HEAD (+ Coordinated word): The system at the moment only accepts coordinated words in nouns and adjectives, so here it is just in case the system is expanded in the future.

g. If H is a Default value from the pattern:
    i. PREPOSITION: If any.
    ii. DEFAULT VALUE.

3. The module also treats the special case of some question particles, like "quant" (*how many*), that need to agree in gender and number with the head of the Theme.

#### 4.5.4. Articles' module

This module will add all the necessary articles to the sentence and apostrophise them if it needs be. The part that puts definite articles and that apostrophises them, if necessary, is a module on its own. This sub-module, as opposed to others that are tailored for the constraint grammar of this system, can be used for any of the words in Catalan as it takes into account all the rules and exceptions to apostrophise nouns in Catalan.

Keep in mind that in most of the cases, articles can either be definite or indefinite for the same noun. Each of the articles can give a small variation to the sense of the sentence. We could have added a word modifier that would be applied from the input that could permit the user to define if the article attached to the noun was definite or indefinite. Nevertheless, as in AAC applications saving as many selections as possible is very important, we decided not to implement this solution. Therefore, what we try to do in this module is to reflect the most common occurrences for the articles in a given situation in natural language, taking into account both syntactic and semantic features.

The module works as follows:

1. For each noun (N) of each slot, the generator determines which type of article it needs:
    a. If N has a quantifier or is a pronoun, then no article is attached to it.

b. Else if N is a complement of another noun, in our system N does not have any article attached, unless N is of class “object”. If it of class “object” then it has a definite article. E.g. “L’home de ferro” (*The man of iron*) versus “La pota de la taula” (*The leg of the table*).
c. Else if N has a possessive, the article is always definite.
d. Else, it depends on the type of slot:
    i. If it is a Subject, the article is definite unless:
        1. The type of sentence is an order. Then there is no article. E.g. “Infermera vagi a l’habitació” (*Nurse, go to the room*).
        2. N is of the location class. Then the generator looks at the annotated features of N in the database and puts the type of article stated there.
    ii. If it is a Location slot, then the generator also looks at the annotated features of N in the database and puts the type of article stated there.
    iii. If it is a Theme:
        1. If there is the question particle “quant” (*how many*), N has no article. E.g. “Quantes patates vols?” (*How many potatoes do you want?*).
        2. If the sentence is an answer and N is “human”, the article is definite. “La mare” (*Mother*).
        3. Else, the generator looks at the annotated features of N in the database and puts the type of article stated there.
    iv. If it is any other type of slot, then the generator looks at the annotated features of N in the database and puts the type of article stated there.

2. Once the module has decided the type of article for each noun:
    a. If it is an indefinite article, it puts the one that agrees in gender and number with the noun (“un”, “una”, “uns”, “unes”).
    b. If it is a definite article, it calls the definite articles’ sub-module that is explained below.
3. Then the generator places the article before the noun or before the possessive, if the noun also had a possessive. E.g. “La casa” (*The house*); “La meva casa” (*My house*); “La meva primera casa” (*My first house*).

| Conditions | | Algorithm with examples | | |
|---|---|---|---|---|
| **Id** | **Condition text** | **Level** | **Condition > Operator** | **Example** |
| a | the word is a noun in the sentence | 0 | a ( | |
| b | the word is in plural | 1 | b ( | |
| | | 2 | d > R | conills [rabbits] -> els conills |
| c | the word begins in consonant except for “h” | 2 | ¬d > S ) | festes [holidays] -> les festes |
| d | the word is masculine | 1 | ¬b ( | |
| | | 2 | c ( | |
| e | the word begins in “a”, “e”, “o” or “ha”, “he”, “ho” | 3 | d > O | molí [mill] -> el molí |
| | | 3 | ¬d > P ) | cadira [chair] -> la cadira |
| f | the word belongs to the lists B, D or F | 2 | ¬c ( | |
| | | 3 | e ( | |
| g | the word begins in “i”, “u” or “hi”, “hu” | 4 | f > P | ema [em] -> la ema |
| | | 4 | ¬f > Q ) | abella [bee] -> l’abella |
| h | the “i” or “u” is unstressed | 3 | ¬e ( | |
| | | 4 | g ( | |
| i | the “i” or “u” is non-vocalic | 5 | h ( | |
| | | 6 | d ( | |
| j | the word belongs to the lists A or E | 7 | i & m > O | iogurt [yoghurt] -> el iogurt |
| | | 7 | ¬ (i & m) > Q ) | ignorant [ignorant] -> l’ignorant |
| k | the word belongs to the list G | 6 | ¬d > P ) | universitat [university] -> la universitat |
| | | 5 | ¬h ( | |
| l | the word belongs to the list C | 6 | j > P | u [u] -> la u |
| | | 6 | ¬j ( | |
| m | the word is not in the list H | 7 | k ( | |
| | | 8 | d > O | hippy [hippy] -> el hippy |
| | | 8 | ¬d > P ) | Harriet [Harriet] -> la Harriet |
| | | 7 | ¬k > Q ) ) | illa [island] -> l’illa |
| | | 4 | ¬g ( | |
| | | 5 | l > Q | 11 [11] -> l’11 |
| | | 5 | ¬l > O ) ) ) ) ) | 300 [300] -> el 300 |
| | | 0 | ¬a > exit | |

**Table 3: Definite articles’ module algorithm: Conditions, algorithm and examples**

#### 4.5.4.1. Definite articles' sub-module

This module has been built following the micro-systemic approach described in Cardey and Greenfield (2008). The specific rules to apostrophise articles in Catalan can be found in appendix F. The algorithm is described in Table 3, the lists necessary for the algorithm are in Table 4 and the operators used are in Table 5.

| Lists | |
|---|---|
| **Symbol** | **Set** |
| A | {una, ira} |
| B | {host} |
| C | {1, 11} |
| D | {a, e, o, efa, ela, ema, ena, erra, essa} |
| E | {i, u} |
| F | {hac} |
| G | {hippy, Harry, Harriet} |
| H | {ió} |

Table 4: Lists of the algorithm

| Operators | |
|---|---|
| **Symbol** | **Operator** |
| O | el |
| P | la |
| Q | l' |
| R | els |
| S | les |

Table 5: Operators used in the algorithm

### 4.5.5. Verb conjugator module

This module conjugates the verbs of the sentence according to their subject and the verb tense of the sentence, which can be given by the type of sentence, a time expression, a tense modifier or by the default tense encoded in the pattern.

1. First, the module determines the tense of the sentence. It picks the default tense of the pattern and checks if it needs to replace it:
    a. If the sentence is an order, if it is positive the tense is imperative, if it is negative, then the tense is present subjunctive.
    b. Else, if the tense has been specified in the input, the module takes that tense.
    c. Else, if there are time expressions that modify the tense, the module takes them into account. E.g. "Ahir vaig anar a comprar" (*Yesterday I went shopping*). "Yesterday" changes the sentences to past tense.
2. Once the tense has been determined, the module starts conjugating all verbs in the sentence in the following way:
    a. If "Wish" or "Permission" modifiers were chosen, verbs "voler" (*to want*) or "poder" (*can*) are the first to be conjugated.

b. Then the main verb is conjugated according to the tense taken and the person and number of the subject.
    i. If the “Permission” modifier was chosen, the main verb, and the secondary verb (if there is any), is in the infinitive form.
    ii. If the “Wish” modifier was chosen, the main verb is in the infinitive form and the secondary verb (if any) goes in the present subjunctive form and agrees with Subject 2 in person and number. When there is a “Wish” modifier, the system takes the subject of the main verb also as the subject of the “voler” (*to want*) verb.
    iii. If there is not any modifier, the main verb is conjugated normally. Then, if there was a secondary verb, if its subject was the same as the main verb, it goes in infinitive. Else, it goes in subjunctive and agrees in person and number with Subject 2. E.g. “Vull menjar” (*I want to eat*); “Vull que em donis la mà” (*I want you to give me your hand*). In this second case, the system also adds the conjunction “que” (*that*).

### 4.5.6. Cleaning module

This last module does the final changes to the generated sentence and gets it ready to output. The changes that it does are the following:

1. Reduce the subjects with the pronominal forms “jo” (*I*) and “tu” (*you*).
2. Transform the receivers in pronominal forms into the correct pronominal form depending on the conjugation of the verb. See all the transformations and the correct placing of the resulting pronouns in Table 6.

| | **Before the verb**<br>**Infinitive or Positive Order** | **After the verb**<br>**Else** |
|---|---|---|
| **jo / mi** | em | me |
| **tu** | et | te |
| **ell / ella** | li | li |
| **nosaltres** | ens | nos |
| **vosaltres** | us | vos |
| **ells / elles** | els | los |

**Table 6: Transformations of feeble pronouns in Receiver slots**

3. Add sentence modifiers that do not go at the beginning of the sentence, which were already added by the first module of the generator. Mainly “no” (*No*), for negative sentences, and “també” (*too*). These modifiers go after the subject and

before all feeble pronouns that could appear before the verb, e.g. “La Maria no ho tenia” (*Mary did not have it*). If there is a sentence modifier of “Wish”, the modifier will go before the verb “voler” (*to want*), e.g. “No vull dormir” (*I don’t want to sleep*).

4. Join prepositions with articles (see Table 7). The generator does it with the use of regular expressions.

| Preposition+Article | Contraction |
|---|---|
| de+el | del |
| de+els | dels |
| a+el | al |
| a+els | als |
| per+el | pel |
| per+els | pels |

**Table 7: Preposition plus article contractions**

5. Join feeble pronouns among themselves and with verbs with apostrophes and hyphens (see Table 8). Also put apostrophes to prepositions. The generator does it with the use of regular expressions.

| Combination | Transformation |
|---|---|
| de+[vowel\|h] | d’ |
| em+[vowel\|h] | m’ |
| et+[vowel\|h] | t’ |
| [vowel]<END-OF-VERB>+me | ‘m |
| [vowel]<END-OF-VERB>+te | ‘t |
| [vowel]<END-OF-VERB>+li | -li |
| [vowel]<END-OF-VERB>+nos | ‘ns |
| [vowel]<END-OF-VERB>+vos | -us |
| [vowel]<END-OF-VERB>+los | ‘ls |
| [vowel]+r<END-OF-VERB>+me | -me |
| [vowel]+r<END-OF-VERB>+te | -te |
| [vowel]+r<END-OF-VERB>+li | -li |
| [vowel]+r<END-OF-VERB>+nos | -nos |
| [vowel]+r<END-OF-VERB>+vos | -vos |
| [vowel]+r<END-OF-VERB>+los | -los |
| <END-OF-VERB>+’m+ho | -m’ho |
| <END-OF-VERB>+’t+ho | -t’ho |

| Combination | Transformation |
|---|---|
| <END-OF-VERB>+-li+ho | -li-ho |
| <END-OF-VERB>+'ns+ho | 'ns-ho |
| <END-OF-VERB>+-vos+ho | -vos-ho |
| <END-OF-VERB>+'ls+ho | 'ls-ho |
| <END-OF-VERB>+-me+ho | -m'ho |
| <END-OF-VERB>+-te+ho | -t'ho |
| <END-OF-VERB>+-nos+ho | -nos-ho |
| <END-OF-VERB>+-vos+ho | -vos-ho |
| <END-OF-VERB>+-los+ho | -los-ho |

**Table 8: Feeble pronouns combinations before and after the verb**

6. Add time expressions that go at the end of the sentence, if any.
7. Add set expressions, if any.
8. Add end of the sentence punctuation depending on the type of sentence (declarative, question, exclamation, etc.).

After all the modules are applied, the sentence is ready to be output. The system saves the output sentence and the parse tree in the database and sends the results to the user interface.

The result for the example sentence (1) used previously:

(1) Ahir volia donar l'ós de peluix vermell a la Marta.

(*Yesterday, I wanted to give the red teddy bear to Martha*)

Figure 22 shows another example of the output, now in the user interface for another input sentence. On it, we can see the input words, the output of the final sentence and the output of the parse tree.

The input words are: "meu / germana / gran / no / anar / casa / ahir" (*my* / *sister* / *old* / *no* / *go* / *home* / *yesterday*).

The generated sentence is: "Ahir la meva germana gran no va anar a casa" (*Yesterday my older sister didn't go home*). From it, we can see that the possessive agrees in gender with "germana" (*sister*). Also, the verb has been conjugated in the past tense due to the time expression "ahir" (*yesterday*) and the articles and prepositions have been correctly included.

Finally, the parse tree shows that the subject, "germana", has got two complements, one is the possessive modifier, "meu", and the other is the adjectival modifier "gran" (*old*).

Also, the whole sentence has a negative modifier, “no”. The rest of the parse tree is straightforward, “casa” (*house*) fits the LocTo slot and “ahir” is a time expression.


**Figure 22: Screenshot of an example of a final output of the *compansion* system**

## 4.6. Interface Design Aspects

In this section essential design aspects for the AAC application will be discussed. Interface features, interface design aspects [Reichle & Drager, 2010 and Taylor et al., 2001], voice selection issues and other relevant information that need to be taken into account when developing the final application. Remember that in the frame of this dissertation project, the final user interface adapted to AAC users has not been built and has been left as the main task for future work. Nevertheless, in order to better understand the context of usage of the *compansion* system, we think that it is interesting for us to define as clearly as possible the final user interface (or final input system).

The first relevant feature of the input system will be a pictogram[48] prediction system that will evaluate the frequency of usage of the symbols by the user depending on the previously selected items on the sentence. The system will take into account the relevance of the missing slots that need to be filled according to the already selected

[48] Pictograms used in our AAC application will be the open-source PCS system from ARASAAC.

items (i.e. if a transitive verb is selected, the theme slot will be the most relevant one that needs to be filled). Once the most relevant slots are selected, the vocabulary items that can fit these slots will be filtered by the same features that the semantic parser uses (i.e. if a verb needs human themes, only nouns or pronouns that correspond to humans will pass the filter). Finally, the remaining vocabulary items will be sorted, first by the number of times that the individual uses that word along with the selected verb (relative frequency) and, second, by the total number of times that that word has been used (total frequency). About relative frequency in questions, instead of (or apart from) making it depend on the verb of the sentence, we may also make it depend on the question particle. The total number of predicted pictograms displayed on screen will be between 1 and 7 [Taylor et al., 2001]. This number will depend on the number of items that can fit into a row of the display grid. The input system will also have an option in the menu options to activate and deactivate the prediction system.

The application will also have a virtual (on-screen) keyboard, so that users that have minor disabilities can directly write the text that they want to be synthesised without using the PCS symbol. However, the input from the virtual keyboard will not be processed by the *compansion* system. The virtual keyboard will not have a word prediction system and its input will most probably not be checked for spelling or syntactic errors.

Another interesting feature of the system is that it will also remember the recently generated sentences (grouped into sentences used today, last week and last month), so that they can be used again without having to type them from scratch. This is an interesting feature, because it will allow users to prepare in advance questions, answers or any type of sentences for a certain a conversation. This is similar to when people rehearse sentences for meetings or important events that they have to attend to [Copestake and Flickinger, 1998]. The desired, recently used sentences, will be also able to be saved and grouped into custom folders for later use.

On another hand, it is important to keep in mind that most people are not familiar with AAC technologies, that is why it is very advisable to provide a way for the users of the application to explain to their communication partners how they communicate. Consequently, there will be pre-recorded sentences in the application, which the user will also be able to customise, that will tell others about the communication device and the handicap of the user. Probably there will also be pre-recorded sentences to give

some recommendations that need to be taken into account when talking to an AAC user [Larranz, 2006]. These recommendations would be such as the following:

- Ask the user where you should sit in order to make the communication easier for him.
- Ask clear and precise questions, so that the answers can be concrete.
- Give enough time to answer to the user. Be patient.

Regarding the selection of the synthesised voice, not only it is important to select a good quality voice, but also to keep in mind that the application will need at least two different voices. The first one is the voice that will become the voice of the user and the second one is the voice that the program will use to give indications to the users, the feedback voice [Escoin, 2006]. For example, every time that the user selects a pictogram, the program will read the word corresponding to the pictogram aloud to help the user detect possible errors. It is very important for these two voices to be different to avoid confusion for both the user and the communication partner.

In principle, for the future interface, an already available synthesised high-quality voice for Catalan has already been chosen. This is the open-source Catalan voice, which was developed using the Festival voice synthesiser engine, by TALP[49], a specialized group in NLP of the Universitat Politècnica de Catalunya (UPC), in Barcelona. At the moment, no portable device can run the Festival engine, which is the reason why there is a portable version of the engine called Flite[50]. Unfortunately, no Catalan voices are yet available for Flite.

Therefore, presumably one of the tasks in the development of the interface will consist on the development of a web service that will make the FestCat [Bonafonte et al., 2009] Catalan voices[51] and the Festival engine accessible through the Internet. Naturally, the same web service could also incorporate all the voices developed for Festival up until now (English, French, Spanish, Portuguese and German, among others), depending on the computing capacity and the bandwidth of the server hosting it. Besides the final AAC application to be developed, the web service could also be used by any other application that required a synthesised voice in Catalan. For example, it could be used for giving

[49] Centre de Tecnologies i Aplicacions del Llenguatge i de la Parla (TALP). http://www.talp.cat/talp/ [April 15th, 2012]

[50] Flite. http://www.speech.cs.cmu.edu/flite/ [May 1st, 2011]

[51] Demostració de les veus FestCat. http://www.talp.upc.edu/festcat/ [April 15th, 2012]

voice to an interactive Q&A system, reading Catalan newspapers to visually impaired people, and any other text-to-speech application, or it could even be used for giving voice to characters in videogames for handheld devices, just to name a few. This way, the web service itself would be an important resource to the Catalan NLP community as a whole and to the AAC target audiences.

During the initial months of the project, a prototype of the web service was built. Some tests were run on the server in order to see the latency[52] of the service. The delay of the response of the web service was around 3 seconds for simple sentences. That means that from the instant were the user asks for a sentence to be synthesised (without taking into account the time that the *compansion* system will take, which is expected to be negligible), until it is read aloud by the system, there will be a 3 seconds delay, if no solution is found. AAC users usually take quite a long time to build sentences, so 3 seconds is not a very relevant amount of time, according to the AAC professionals that we met. However, a professional AAC application should not have this delay. Another major problem of using a web service as the voice synthesiser engine is that, in order for the voice to work, the device needs to have permanent Internet connection. That is why other commercial voice possibilities have been explored[53].

In any case, to conclude with the selection of the voice engine topic, the application will be developed following a modular design pattern that will let the voice synthesiser engine module to be independent from the rest of the functionalities of the program. This will allow adding or replacing the voice engine in advanced stages of the project.

Going back to the graphical aspects of the user interface, as the application will target individuals with different types of disabilities (even among similar kinds of disabilities there can be a huge difference in physical capabilities, like sight problems, motor impairments and so on) it is very important for the interface to be extremely customisable. Some of the layout aspects that need to be adaptable to each of the users are size and number of pictograms displayed on screen, colour palette of the application, black-on-white or vice versa, among others.

---

[52] Wikipedia. Latency (Engineering). http://en.wikipedia.org/wiki/Latency_(engineering) [April 15th, 2012]

[53] We have contacted the Acapela and Loquendo companies and asked for their pricing policies. If one of their voice solutions were eventually used, the program will not be able to be free any longer. In any case, further negotiations still need to be done regarding this topic.

Another crucial point is to make the software accessible for people who might use single-button or two-button switches, external keyboards or other peripherals as a substitute for the touch screen of the selected iOS device. Although, in the scope of the final project, we will not worry about the available peripherals and how they can be connected to the devices, we need to at least ensure that, once they are correctly plugged, the user interface can be configured for its use. That specially implies implementing different types of scanning methods [Lesher, 1998] to access the items on the user interface.

The most basic scanning method is linear scanning, where the cursor advances from item to item one-by-one and this way sweeps the entire screen. Then there is row or column scanning, where it advances row-by-row or column-by-column. When the user selects a certain row or column, linear scanning is usually applied to the selected subgroup, though further row scanning can also be successively applied. Finally, there is group scanning, that works exactly like row scanning, but instead of rows, the cursor advances by groups or clusters of items (square or rectangle shape blocks). The three scanning methods can advance through automatic scanning; where there is a set customisable speed that makes the cursor advance and the user just needs to select the desired item when it is highlighted; or through manual scanning; where the user controls the cursor speed by pressing a button every time that he/she wants the cursor to advance. Then to select an item, the user presses another button, on multiple-button switches, or presses the same button during a longer period of time, on single-button switches. This two-button switch behaviour can be mimicked in touch screens using one of two fingers or splitting the screen in two, emulating the right and left click of a mouse.

Ideally, all these scanning methods, plus the customisation of all the scanning parameters (cursor speed of automatic scanning, number of items per scanning group for group scanning, required holding time for single-button or single-finger manual scanning, etc.), should be implemented on the application.

Going back to the configuration and distribution of the boards, it is necessary to remind that each user has different needs in terms of vocabulary, clustering of the different pictograms into groups, size of the images and so on. That is why a basic initial configuration will be provided in order to showcase how the application works: Nevertheless, everything, except for some structural items, will be fully customisable. The only fixed items will be:

- The grid structure where the pictograms will be distributed creating different boards. If all the pictograms in a certain board do not fit the screen, the board will be split into pages. The number of boards consecutively embedded should not exceed 2 or 3. Higher values make board navigation a tiresome task.
- An upper field where the selected items that will construct the sentence will appear.
- Another field, under the sentence field, with 5 to 7 slots for pictograms, where the pictograms chosen by the prediction system will appear.
- A menu for navigating through the boards, erasing the last added symbol, erasing the whole sentence, speaking the sentence aloud, accessing the options area, the recently used sentences, the virtual keyboard and more.

The process of customisation needs to be as user friendly as possible, as most of AAC users and facilitators do not have a technical background. That is why it was decided that the largest part of the customisation will most probably be done through a web page. The main advantages of this decision are:

- The user, if he/she wants to, will be able to work with a bigger screen than if the process were exclusive to their chosen portable device.
- As the screen will be bigger, more pictograms will be able to fit into it and it will be easier to create the custom boards.
- Normal users are more familiar with computers than with touch screens.
- It will be easier to write down long custom presentation messages from the keyboard of the computer.

Main disadvantage:

- If the user usually runs the application from a portable device, the device will need to be plugged to the computer and a configuration file will need to be transferred to the device.

In principle, the rest of the customisation (the size of the icons, whether they want to enable or disable the pictogram prediction system, enabling or disabling the scanning system and selecting the desired scanning method, etc.) will be done on the application from the device.

As for the implementation, as stated before, the application will be built following a modular design that will allow for future vocabulary updates, like adding person names for the relatives and friends of the user, or voice upgrades.

### 4.6.1. Device selection

As seen in the Introduction and in the State of the Art review, the AAC scene at the moment is switching towards portable devices that enable CCN individuals to use them in any communicative situation without any previous setup, instead of computers that need to be wired and laptops, which are not as portable as they may appear and that take a long time to boot and be ready to use.

Tablets, apart from being highly multi-functional, bundle software and hardware to fulfil all the major communication needs (surfing the net, emailing, text messaging, accessing social networks, gaming, reading, listening to music, calling, etc.) and also offer good interoperability. They can connect and communicate to other computers and other devices through Wifi, Bluetooth and USB connectors. Moreover, some AAC peripherals are already being designed specifically for tablets[54].

Among mainstream smartphones and tablets, iPhone, iPad and iPod Touch have been chosen for the final application since they all share the same developing system (iOS) and, by their characteristics, mainly variety in screen size, they can reach a wider audience of potential AAC users. iOS devices are also the leading in sales devices of their kind[55]. Nevertheless, if the core of the application is found online, like the *compansion* system is at the moment, the software will also be accessible from laptops, desktop computers and any other devices that have Internet connection.

[54] RJ Cooper & Associates, Inc. http://www.rjcooper.com/ [April 15th, 2012]

[55] http://www.mobileburn.com/news.jsp?Id=13194 & http://www.bit-tech.net/news/hardware/2011/05/16/nvidia-ceo-not-pleased-with-android-tablet/1 [May 1st, 2011]

# 5. Evaluation

In order to evaluate the system, we gave a set of 100 natural language sentences to 3 different annotators (a linguist, a translator and a computer scientist). The annotators had to input these 100 sentences and then evaluate separately the output of the parser and the final output of the generator. With this, our aim was:

- To evaluate if the system could take different inputs for the same target sentence, which was one of the goals of the project.
- To evaluate the two main components of the system, the parser and the generator.

But before detailing the obtained results, let us better explain each of the steps of the evaluation process.

## 5.1. Selection of the set of sentences for the test

We contacted two of the associations that we had previously visited on the first year of the project, to better understand how pictogram-based AAC system work and to see if they could provide us with a representative set of sentences to test our system. Answering our request, one association sent us a set of 120 basic sentences that they commonly use to teach and train new AAC users on the use of their AAC panels or devices.

As our system at the moment has reduced vocabulary, mainly limited verbs and verb senses available, we had to adapt the initial set. Basically, we removed the sentences that used verbs that our system does not have or we substituted them with verbs that had a similar sentential structure (in total 30 sentences). For example, we removed sentences like “Se m’ha trencat la cadira de rodes” (*My wheelchair got broken*) and replaced other sentences like “Puc telefonar, si us plau?” (*Can I phone, please?*), for “Puc jugar a pilota, si us plau?” (*Can I play ball, please?*). We also substituted missing nouns and adjectives and replaced them by words that our system has in its database.

Concerning complex sentence structures, like subordinate clauses or chains of noun complements, we did not have to remove any of them, which confirmed that, although our system has limitations in this areas, as previously explained in section 4, it is sufficient for the final users’ needs, at least at this stages. Regarding other sentence structures, we only had to remove five sentences that used verbs that we had in the

system, but with other senses, like sentences that talk about the weather (i.e. “Fa sol” – *It’s sunny*), which our system does not support at the moment, but which can be easily added. Also, we had to remove a couple of sentences that used question particles that we do not have at the moment (i.e. “Amb qui vas a la festa?” – *Who are you going to the party with?*).

Finally, as there were many similar sentences in structure (i.e. “Estic cansat”, “Estic content”, etc. – *I’m tired*, *I’m happy*, etc.) and not many complex sentences, we added some sentences that tested some of the most complex features of the system, like the coordination of two nouns or two adjectives, more sentences with two verbs in them and sentences with nouns acting as noun complements.

The final set of 100 sentences can be found in appendix G.

## 5.2. Evaluation of the input

The input system has not been our concern during this project, as the final input system will be very different from the existing one. The one that has been built is not adapted to the end user at all, as it has the only purpose to feed the *compansion* system. Even though, there is an aspect of the input that is interesting to evaluate because it can have a direct effect on the parser and, therefore, on the generated sentence. That is the different ways in which the users can enter the words (or pictograms) to build the same target sentence.

To see this, we divided all sentences input by the annotators in the following three groups:

1. Sentences that all annotators input in the same way
2. Sentences that one annotator did differently
3. Sentences that all three annotators input differently

Then, to see the main differences in the input, we classified the sentences that were input differently in the seven following categories:

1. Subject omitted
2. Receiver omitted
3. Possessives input in a different order
4. Quantifiers input in a different order

5. Different sentence modifiers applied[56]
6. Temporal expressions input in a different order
7. Other words inputted in a different order

If a sentence had two of these problems, in the evaluation of the results, we counted the sentence as half in each of the categories.

Finally, to compare and see if the results varied a lot if the sentences where built differently or not, we classified equally built sentences and differently built sentences separately into two categories:

1. Good sentences (the sentence is perfect or it can be understood although there are minor generation issues)[57]
2. Bad sentences

If we found sentences that one annotator classified as good and the rest as bad or viceversa, we counted the sentence as half in each of the categories.

We also annotated sentences that were differently built but had exactly the same output generated sentence.

## 5.3. Evaluation of the parser

To evaluate the parser, the annotators had to classify the output of the parser for each of the input sentences into 4 different categories:

1. The analysis of the parser is correct.
2. The analysis of the parser is not the expected, although the solution reached could also be correct, as it does not change too much the meaning of the sentence.
3. The analysis of the parser is not correct: there is a single parsing error.
4. The analysis of the parser is not correct: either there are several errors or the parsing does not make any sense at all.

[56] Category number 5 stands for different verb tenses applied (i.e. leaving the system to choose the default tense for that given sentence or specifying manually the verb tense) and for different sentence type modifiers (declarative, question, order, answer, wish, etc.).

[57] We will better detail this classification when we talk about the evaluation of the generator in section 5.4.

Moreover, as we wanted to have more details on the errors made by the parser, category 3 was further subdivided into several subcategories (Figure 23, page 74):

a. Error in the detection of the subject.
b. One of the adjectives is not parsed in the correct place.
c. Error in the detection of a noun as a noun complement.
d. The parser has misplaced a noun in a slot where it should not go.
e. The parser has chosen a verb sense, which is not the desired one.
f. The parsed sentence has an error that can be derived from the word order in the input.

Also, to evaluate the interannotator agreement on the parser, we calculated Randolphs' free-marginal multirater kappa (Randolph 2005; Warrens, 2010) using the Online Kappa Calculator[58]. Brennan and Prediger (1981) suggest using free-marginal kappa when raters are not forced to assign a certain number of cases to each category, which is the case for our project, as annotators could freely grade each of the sentences separately. That is why we selected a free-marginal kappa over a fixed-marginal kappa. The values of kappa can range from -1.0 to 1.0:

- -1 would indicate perfect disagreement below chance.
- 0 would indicate agreement equal to chance.
- 1 would indicate perfect agreement above chance.

A general rule is that a kappa of 0.70 (or above) indicates adequate interannotator agreement.

To calculate the kappa we took into account categories 1-4 explained above, without considering the different subcategories in 3.

Finally, we also wanted to have an overall score for the parser, so we decided to translate the qualitative scale of categories 1 to 4 into a balanced quantitative scale 0-10, using the following conversion:

- Category 1 would evaluate as a 10.
- Category 2 would evaluate as a 7.
- Category 3 would evaluate as a 3.
- Category 4 would evaluate as a 0.

[58] Randolph, J. J. (2008). Online Kappa Calculator. Retrieved April 17, 2012, from http://justus.randolph.name/kappa. [April 17th, 2012]

Like this, if each of the categories had the same number of selections, the final average score of the parser would be a 5.

## 5.4. Evaluation of the generator

To evaluate the generator, the annotators had to use a qualitative scale similar to the one used with the parser. The four categories in which the annotators had to classify the generated sentences were the following:

1. The sentence is perfectly generated.
2. The sentence is well generated and it can be understood, although there are some minor problems.
3. The sentence is well generated, but it cannot be understood, as there are errors than come from the parser.
4. The sentence is wrongly generated and cannot be understood.

Furthermore, like it has been done before with the parser, in order to gather more information on the issues of the generator, categories 2 and 4 are further subdivided. Category 2 has the following subcategories:

a. There are minor errors in the word order of the generated sentence.
b. There are minor errors in the choice of the articles (mainly definite vs. indefinite) of the generated sentence.
c. There are minor errors with the conjugation of the verbs.
d. There are other minor errors that do not come from the parser, which are not described in a-c, above.
e. There are minor errors that come from the parsing.

And category 4 has the following subcategories:

a. The sentence is wrongly generated and cannot be understood, although the parsing was correct.
b. The sentence is wrongly generated and cannot be understood and the parsing was also incorrect.
c. The sentence was not generated at all, due to an internal error of the system.

As with the parser, we also calculated Randolphs' kappa to assess the interannotator agreement when rating the output of the generator. To do so, we used categories 1-4 without taking into account the subcategories of neither 2 nor 4.

Finally, to have an overall score of the performance of the generator, we transformed the qualitative scale of categories 1 to 4 into a balanced quantitative scale 0-10, using the same conversion that we used with the parser:

- Category 1 would evaluate as a 10.
- Category 2 would evaluate as a 7.
- Category 3 would evaluate as a 3.
- Category 4 would evaluate as a 0.

To simplify the analysis of the results, in the following sections, when we talk about “Good sentences” and “Bad sentences”, we considered that good sentences are the ones that fall into categories 1-2, which are either perfect or can be understood; while bad sentences are the ones in categories 3-4, which cannot be understood.

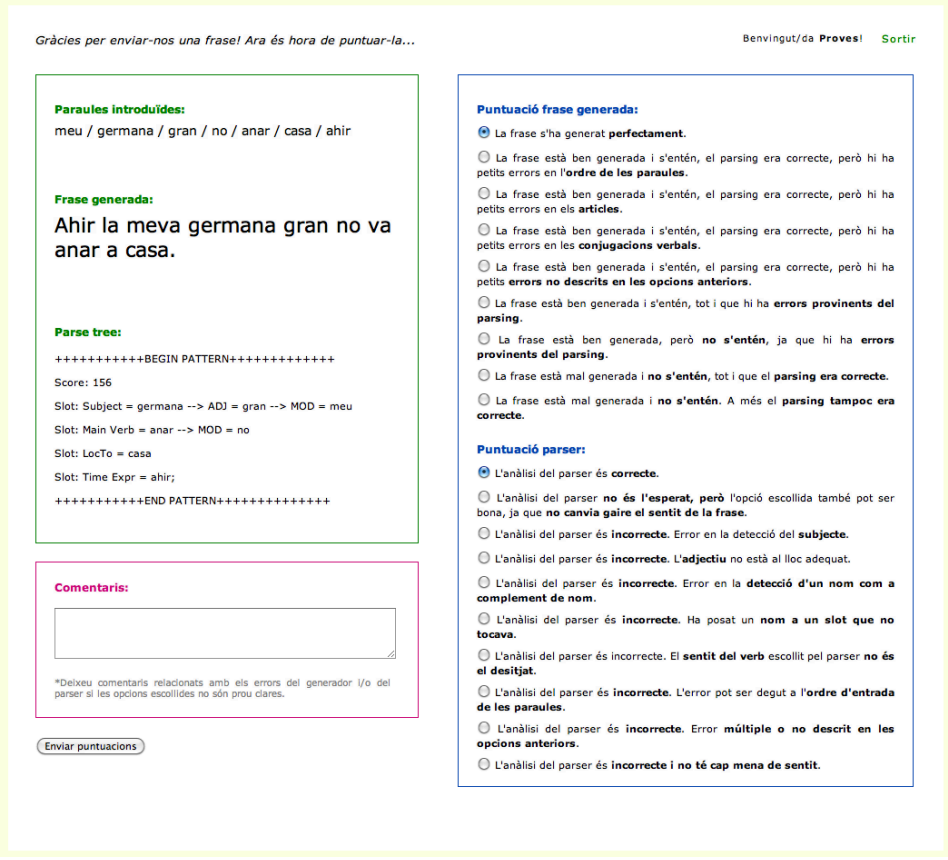



**Figure 23: Screenshot of the output and the evaluation form**

## 5.5. Results of the evaluation of the input

The raw data with all the input sentences by the annotators, along with their respective results and rates of the parser and generator, can be found in appendix H.

Having noted this, we will start the analysis of the input of the 3 annotators by looking at the results in Table 9.

| | |
|---|---|
| % Sentences that all annotators inputted the same | 40% |
| % Sentences that one annotator inputted differently | **50%** |
| % Sentences that all annotators inputted differently | 10% |

**Table 9: Differences in the input of the sentences by the annotators**

Table 9 shows that only 40% of the sentences have been input using the exact same words (in the same order) and the same modifiers by all annotators. This confirms one of the concerns in the development of the system, which was that it had to be flexible enough so that it could treat different inputs for the same target sentences. In Table 11, we can see that this problem has not translated in the final results, as 97,5% of the sentences are well generated, even when the input by the annotators was different, against a 100% of the sentences that are good when the input is the same.

In Table 10, we can see that, by far, the most common difference is the omission of the subject in the sentence (which amounts to a 50% of the total differences in the input). Another important difference, with a 25% of occurrences, is the different use of modifiers in the sentence. Taking a closer look at these sentences, we can see that it is mostly due to the use of the verb “voler” (*to want*) instead of the modifier “Desig” (*Wish*), the use of the word “No” (*No*) instead of the modifier “Negativa” (*Negative*) and the use of the modifier “Resposta” (*Answer*) in short sentences. As stated before, we can see in Table 11 that all these differences have been, in general, correctly treated by the system.

Another interesting data that reinforces this last statement is that a 91,67% of the sentences that were input differently by the annotators have exactly the same output. In other words, a 95% of the total outputs by the generator (both outputs that came from sentences that had the same input, which are always identical, and outputs of sentences that had a different input) are identical.

| | |
|---|---|
| % Omission of the subject | **50,00%** |
| % Omission of the receiver | 0,83% |
| % Different order of the possessives | 5,00% |
| % Different order of the quantifiers | 7,50% |
| % Different sentence modifiers used | 25,00% |
| % Different order of the time expressions | 5,83% |
| % Different order in other words | 5,83% |

**Table 10: Types of differences in the input**

| | |
|---|---|
| % Good sentences when input was the same | **100,00%** |
| % Bad sentences when output was the same | 0,00% |
| % Good sentences when input was different | **97,50%** |
| % Bad sentences when input was different | 2,50% |
| % Identical output when input was different | 91,67% |
| % Good sentences in total | **98,50%** |
| % Bad sentences in total | 1,50% |
| % Identical output in total | 95,00% |

**Table 11: Good, bad and identical sentences depending on the type of input**

Other general observations that we could extract from the input sentences are that there is a small learning curve to use the system and take advantage of its capabilities. We observed that, in the beginning, most of the annotators stated all the words (including all the redundant subjects “I”, which the system usually takes as default), while in the middle or at the end of their task they only stated the subject in sentences where there could be an ambiguity. The same happened in the use of sentence modifiers. In the beginning, annotators would rather use the verb “voler” (*want*), instead of the modifier “Desig” (*Wish*), or they did not use the modifier “Resposta” (*Answer*) to specify that the

sentence could very well be “verbless”, while at the end, annotators used modifiers at its fullest.

Another indicator that there is a small learning curve is that, in the beginning, annotators repeated some of the sentences (the most complex ones, in terms of parsing complexity) once or twice to get the desired result, while afterwards, this was reduced to minimum. For example, the sentence “Un got d’aigua, si us plau” (*A glass of water, please*) was entered twice by one annotator: 1) got / aigua / si us plau (without modifiers) and the output was “El got és aigua, si us plau” (*The glass is water, please*); 2) got / aigua / si us plau (modifier: “answer”) and the output was the desired “Un got d’aigua, si us plau” (*A glass of water, please*). There was another annotator that had the same output as 1), but that did not repeat the sentence. However, on the next similar sentence, the annotator used the modifier “answer”.
Besides, what confirms that it is a “small learning curve” and not a bigger one, is that after having input around 70 sentences, all annotators seldom had to repeat the input to obtain the desired result.

All the detailed results for the input, sentence by sentence can be found in appendix J.

## 5.6. Results of the parser

As we can see in Table 13, there has been nearly no disagreement among the annotators (the free-marginal Kappa is 0,942) when grading the parser. Even more, the amount of outputs of the parser that were labelled as correct parses (Category 1) by all annotators goes up to 94%.

We can also see from Table 12, that the subcategories to better detail the errors of the parser in Category 3 have been barely used. The total number of sentences that were classified as good parses (Category 1 or 2) is 98,33%.

| | |
|---|---|
| % Sentences rated as Category 1 | **97,33%** |
| % Sentences rated as Category 2 | 1,00% |
| % Sentences rated as Category 3 | 0,33% |
| % Sentences rated as Category 4 | 1,33% |

**Table 12: Rates of the parser outputs**

| | |
|---|---|
| Average Score of the parser (0-10) | **9,81** |
| % Sentences rated Category 1 by all annotators | 94% |
| % Sentences rated Category 1 or 2 by all annotators (good parses) | **96%** |
| Percent of overall agreement | 0,957 |
| Free-marginal Kappa | **0,942** |

**Table 13: Parser average score, % of good parses and Kappa**

Also, if we look at the detailed rates for the parser sentence by sentence, which can be found in appendix K, we can see that there was not a single sentence rated as Category 3 or Category 4 by all annotators. All these disagreement comes from sentences that were input differently by the annotators. In all cases, there was at least one annotator that found a way to produce a good sentence. This leads us to think that the parser is indeed ready to support the type of sentences for which it was conceived.

## 5.7. Results of the generator

In the generator assessment, there has been a little bit more disagreement among the annotators, although the Kappa is still clearly above 0.70. As shown in Table 16, Kappa is 0.813, which still indicates good interrater agreement. Here, also as opposed to in the assessment of the parser, the number of sentences rated in Category 1 (perfectly generated sentences) by all annotators has dropped to a 74% (Table 16). Nevertheless, the number of sentences rated either in Category 1 or 2 by all annotators is a near perfect 97%.

| | |
|---|---|
| % Sentences rated as Category 1 | **86,33%** |
| % Sentences rated as Category 2 | 12,33% |
| % Sentences rated as Category 3 | 1,00% |
| % Sentences rated as Category 4 | 0,33% |

**Table 14: Rates of the generator outputs**

In Table 15 below, we can see the types of minor errors that the annotators pinpointed in generated sentences that fell under category 2. If we further subdivide some of the types

of errors found going sentence by sentence (see appendix H, L and M), we observe the following:

- All the word order issues are due to moving time expressions at the beginning or at the end of the sentence (66,67% of cases), when it should be the opposite (the correct order depends on the specific time expression). For example, the generated sentence “Nosaltres anirem a cantar el dimecres” (*We are going to sing on Wednesday*), would have been better generated as “El dimecres nosaltres anirem a cantar” (*On Wednesday, we are going to sing*). Another difference occurs when the generator puts the subject before the verb in “pseudo-impersonal” verbs like “agradar” (*to like*), which goes usually after the verb (i.e. “Les pomes m’agraden”, should be better generated as “M’agraden les pomes” (*I like apples*)). Nevertheless, even though the word order might not be the usual in these two cases, both result in perfectly well formed sentences in Catalan.
- Nearly half of the occurrences of other issues (44,44%) are due to pronominal subjects that could have been omitted, but that the generator did not reduce (“ell”, “nosaltres”, “vosaltres” – *he*, *us*, *you* (plural)). This was a decision made taking into consideration the context of AAC users: redundancy of the subject helps avoid ambiguity and reinforces the role of the addressee who the AAC user is talking about.

| | |
|---|---|
| % Subcategory 2.a (word order) | 23,10% |
| % Subcategory 2.b (article) | 26,92% |
| % Subcategory 2.c (verb conjugation) | 11,54% |
| % Subcategory 2.d (other) | **34,62%** |
| % Subcategory 2.e (parsing) | 3,85% |

**Table 15: Types of minor errors in the generator outputs**

Another issue that can slightly alter the meaning of the generated sentence is the inadequate selection of articles, which totals a 27% of the occurrences of minor generation problems. Improving the algorithm that selects the best article in context for a given noun will remain as a task left for future work.

| | |
|---|---|
| Average Score of the generator (0-10) | **9,53** |
| % Sentences rated Category 1 by all annotators | 74% |
| % Sentences rated Category 1 or 2 by all annotators (good sentences) | **97%** |
| % Sentences rated Category 1 or 2 in total (good sentences) | **98,66%** |
| Percent of overall agreement | 0,860 |
| Free-marginal Kappa | **0,813** |

**Table 16: Generator average score, % of good parses and Kappa**

Finally, to conclude the analysis of the results of the generator, just add that we do not calculate the overall F-score in our system, as when precision and recall are the same, which is our case (all inputs had an output), the F-score is the same as the percentage of good sentences. As we can see in Table 16, in the generator this measure goes up to 98,66%.

All the detailed results for the generator can be found in appendix L, along with the detailed results for the parser.

# 6. Conclusion

After working for nearly two years in this project, it is rewarding to have obtained good results. Even though, the full project is far from over, we definitely achieved the goals set for the first part, which entailed the highest degree of difficulty in terms of natural language processing. This encourages us to continue with the rest of it, knowing that its core is solid.

In this project we have built a *compansion* (compression-expansion) system that expands telegraphic language (utterances with only uninflected content words) into natural language sentences in Catalan, in the context of an Augmentative and Alternative Communication (AAC) software. The main two components of the system are: a syntactic-semantic dependency parser and a generator that constructs the final sentence.

At the beginning of the project, we visited three associations of AAC users, which set the starting point to understand the depth of AAC and the needs of its users, which would eventually translate into decisions to make regarding the *compansion* system. With these visits, we were able to confirm that a system that expanded telegraphic utterances into natural language sentences would be an improvement in the quality of life of AAC users.

The first decision that we took was concerning the vocabulary that we would use in the system. Due to the complexity of building a vocabulary from zero, we decided to use the existing AAC vocabulary named CACE. CACE is a basic vocabulary, with over 800 items, specially designed for daily life activities, which can be integrated in commercial AAC systems like The Grid 2, Speaking Dynamically or Plaphoons.

We reduced the initial vocabulary to 600 items and enriched it by syntactic and semantic information in view of the parser and the generator. We also added some sentence type modifiers to the initial set to allow for more flexibility when building the input sentence. Another one of the most important parts of this enriched information are the patterns that we encoded for each of the verbs present in the vocabulary. These patterns describe the sentence structures that each of the verb senses allow.

Then, we built a prototype input system, which is not suitable for AAC and which will not be in the final application, but so that we could feed the rest of the *compansion* system with the vocabulary.

Afterwards, we designed the parser. The parser accepts a reduced vocabulary and uses a controlled grammar, since it would not be possible to accept all the type of sentences present in natural language. The parser extracts sentence patterns taking into account input words (basically verbs present in the input) and their modifiers. It then assigns the remaining words to the structural slots of those patterns, which correspond to the semantic roles of the predicative elements and of other words' complements. To do so, the parser mainly uses the semantic information encoded in the words and in the verb patterns. It only uses syntactic information (i.e. language-dependent word order) to disambiguate between words that can fit the same slot, when semantic information is not enough. It also uses syntactic information for nouns that act as noun complements, which the system restricts to nouns that immediately follow other nouns in the input. This makes the parser nearly language independent. Adapting it to other languages, like Spanish, would just involve minor changes to the code (apart from the tagging of all the vocabulary).

Then, we built the generator that constructs the natural language sentence, from the resulting parse tree. The generator is formed by several independent modules that progressively expand the parse input step by step. First, there is a slot-ordering module that reorders the components of the sentence according to the sentence type. Then, a word order module, which orders the words within the slots and which adds prepositions and makes sure that words agree in gender and number with each other. Afterwards, the articles' module, which attaches articles to nouns, if necessary, followed by the verb conjugator module. Eventually, there is a cleaning module that deals with the final details and gets the sentence ready to output.

Finally, after building the *compansion* system, we asked three independent annotators to conduct the tests and to score the system using a set of 100 sentences. The 100 sentences were adapted from a set of 120 that one of the associations that we visited uses to teach and train new AAC users on the use of their AAC panels or devices. The annotators had to input these 100 sentences and then evaluate separately the output of the parser and the final output of the generator. With this we wanted to evaluate both, if the system could take different inputs for the same sentence and build a correct output, and the two main components of the system, the parser and the generator. By the

figures obtained from the three independent annotators, 98,7% of the generated sentences are correct.

Summarising what we achieved, we were able to build a *compansion* system for Catalan. It is obviously in the scope of a controlled grammar, but the system is able to produce quite complex sentences and, most of all, for the structures that it supports, it is reliable. Even more, in the context of AAC, the controlled grammar is not a downgrade, as a more complex grammar would probably mean a highest rate of errors by the *compansion* system and a highest degree of difficulty for the users to input the sentence. Keeping in mind the final context of usage of the future application, we think that our controlled grammar should be more than enough in most cases and that our system will be balanced in terms of input difficulty and results. It will also be an improvement over existing AAC software that still produce utterances in the form of telegraphic language. Nevertheless, once the full application is tested, if the grammar needs to be expanded, the system has been designed to do so.

What our system is lacking at the moment is a larger vocabulary. However, the *compansion* system itself does not depend on the vocabulary, as all the rules for parsing and the later generation stage are independently encoded in the system, so that adding new vocabulary is just a matter of annotating the features for each new word, while the code does not need to be changed for this matter.

## 6.1. Future work

The list of tasks to do for future work comes both from features that, due to time constraints, we could not add to the *compansion* system and from issues that can be improved and that we obtained from the analysis of the results in the evaluation phase.

The first, foremost and longest future task is to build the user interface previously described in this dissertation, so that the software becomes a full AAC application. This task also includes annotating the vocabulary that we removed from the initial set of vocabulary in CACE.

The rest of the tasks are small improvements to the *compansion* system. Here is a list of some of them:

1. Split existing verb patterns into several patterns that would be more detailed. It would allow distinguishing between sentences that have the same structure. For example, sentences that when the subject is of the class “object” the theme

needs to be an “object” too, while when the subject is “human” the theme can be any class of “noun”; or sentences that, when the subject is a noun of the semantic class “human” they can have a receiver, while, when the subject is of the semantic class “object”, they cannot.

2. Modify the system so that the same verb pattern can be used for several verbs. This would allow to faster increase the number of verbs available in the system.
3. Add patterns for sentences that express meteorological phenomena, i.e. that talk about the weather, e.g. “Fa sol” (*It’s sunny*).
4. Review all verb patterns to better specify the classes of nouns that each slot prefers.
5. Add more vocabulary. For example, demonstrative (deictic) determiners like “this”, “that”, “here”, “there”, etc.
6. Increase the amount of noun classes and adjective classes so that the patterns could be more fine-grained.
7. Add the feature “ser/estar” (*to be*) to adjectives. It would allow the system to decide which verb goes better with a certain adjective in sentences where the user did not specify the verb in the input. E.g. “Content” → “Estic content” (*Happy → I’m happy*) versus “Sóc content”, which would be incorrect.
8. Add a list of available person names (the user interface should also have the ability to add person names to the system, to add the names of relatives and friends).
9. Add the past imperfect verb tense as the default past tense for some verbs that usually use it over the perfective tense. E.g. “Estava cansat” (*He was tired (imperfect)*) should be the choice over “Va estar cansat” (*He was tired (perfect)*).
10. Modify the system so that the user could specify what time is it. E.g. “Són tres quarts de dues” (*It is a quarter to two*). At the moment, only “on the hour” times can be built.
11. Separate set expressions in two groups, expressions that go at the beginning of the sentence and expressions that go at the end. “Bon dia, vull una poma, si us plau” (*Good morning, I want an apple, please*). At the moment, set expressions are thought to be used as standalone sentence.

The *compansion* system is provisorily available for demonstration and testing at the following url:

http://s384456052.mialojamiento.es/

Username: demo

Password: demo

# 7. References


Augé, C. and Escoin, J. (2003), *Chapter 7: Tecnologías de ayuda y sistemas aumentativos y alternativos de comunicación en personas con discapacidad motora,* in Alcantud, F. (2003), Tecnologías de ayuda en personas con trastornos de comunicación, Nau Llibres.

Blackstone, S. (2009), *AAC Communication and Technology*, Augmentative Communication News (ACN), Vol. 21, Num. 3, pp. 1-8.

Blackstone, S. (2007), *AAC Research and Development: Having an Impact*, Augmentative Communication News (ACN), Vol. 19, Num. 3, pp. 1-4.

Bonafonte, A., Aguilar, L., Esquerra, I., Oller, S. and Moreno, A. (2009), *Recent work on the FESTCAT Database Speech Synthesis*, I Joint SIG-IL / Microsoft Workshop on Speech and Language Technologies for Iberian Languages, 2009.

Cardey, S. and Greenfield, P. (2008), *Micro-systemic Linguistic Analysis and Software Engineering: a Synthesis*, in revue RML6, 2008, Actes du Colloque International en Traductologie et TAL, Editions Dar El Gharb, Oran, pp. 5-25.

Copestake, A. and Flickinger, D. (1998), *Enriched Language Models for Flexible Generation in AAC Systems*. Technology and Persons with Disabilities Conference (CSUN-98). Los Angeles, CA.

Copestake, A. (1997), *Augmented and Alternative NLP Techniques for Augmentative and Alternative Communication*, In Proceedings of the ACL Workshop on Natural Language Processing for Communication Aids, Madrid.

Covington, M-A. (1990a), A Dependency Parser for Variable-word-order Languages, *Technical Report AI-1990-01*, University of Georgia.

Covington, M-A. (1990b), Parsing Discontinuous Constituents in Dependency Grammar. *Computational Linguistics* 16: pp. 234–236.

Covington, M-A. (1994), Discontinuous Dependency Parsing of Free and Fixed Word Order: Work in Progress, *Technical Report AI-1994-02*, University of Georgia.

Covington, M-A. (2001), A Fundamental Algorithm for Dependency Parsing. *Proceedings of the 39th Annual ACM Southeast Conference*, pp. 95–102.

Debusmann, R., Duchier, D. and Kruijff, G-J. (2004), Extensible Dependency Grammar: A New Methodology. *Proceedings of the Workshop on Recent Advances in Dependency Grammar*, pp. 78–85.

Duchier, D. (2003), Configuration of Labeled Trees under Lexicalized Constraints and Principles. *Research on Language and Computation*, 1: pp. 307-336.

Duchier, D. and Debusmann, R. (2001), Topological Dependency Trees: A Constraint-Based Account of Linear Precedence. *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 180–187.

Eisner, J. M. (2000), Bilexical Grammars and their Cubic-time Parsing Algorithms. In Bunt, H. and Nijholt, A. (eds), *Advances in Probabilistic and Other Parsing Technologies*, Kluwer, pp. 29–62.

Escoin, J. (2006), *Tecnología informática en comunicación aumentativa*, Minusval, Diciembre 2006, Madrid, pp. 23-25.

Gaifman, H. (1965). Dependency Systems and Phrase-structure Systems. *Information and Control* 8: 304–337.

Garay-Vitoria, N., & Abascal, J., (2004), *A Comparison of Prediction Techniques to Enhance the Communication Rate*. Lecture Notes in Computer Science. Springer, Berlin, Vol. 3196/2004, pp. 400-417.

Golinker, L. (2009), *Making and Remaking the Case for AAC*, Augmentative Communication News, Dec 2009, Vol. 21, Num. 4, pp. 10-13.

Hays, D-G. (1964). Dependency Theory: A Formalism and some Observations. *Language* 40: pp. 511–525.

Higginbotham, J. (2007) *Access to AAC: Present, Past, and Future*, Augmentative Alternative Communication. Vol. 23, No. 3, pp. 243-257.

Järvinen, T. and Tapanainen, P. (1998). Towards an Implementable Dependency Grammar. *In Kahane, S. and Polguère, A. (eds), Proceedings of the Workshop on Processing of Dependency-Based Grammars*, pp. 1–10.

Karberis, G. and Kouroupetroglou, G. (2002), *Transforming Spontaneous Telegraphic Language to Well-Formed Greek Sentences for Alternative and Augmentative*

*Communication*, Lecture Notes in Computer Science, Vol. 2308/2002, Methods and Applications of Artificial Intelligence, pp. 745-756.

Karlsson, F., (1990). Constraint Grammar as a Framework for Parsing Running Text. *In Proceedings of the 13th Conference on Computational linguistics: the 13th conference.* Helsinki, Finland, pp.168-173.

Kent, R. (Eds). (2007), *Communicative Disorders Review*, Plural Publishing Inc, San Diego.

Kraat, A. (1990), *Augmentative and Alternative Communication: Does it Have a Future in Aphasia Rehabilitation?*, Aphasiology, Vol. 4, Issue 4, pp. 321-328.

Larranz, C.(2006), *La comunicación aumentativa,* Minusval, Diciembre 2006, Madrid, pp. 17-19.

Lesher G., Moulton B., Higginbotham J. (1998), *Techniques for Augmenting Scanning Communication*. Augmentative Alternative Communication, Vol. 14, pp. 81–101.

Makris, P. (2007), *Multilingual Augmentative Alternative Communication System*, In Proceedings of the 2007 COST action 2102 International Conference on Verbal and Nonverbal Communication Behaviours, Springer-Verlag, Berlin.

Maruyama, H. (1990), Structural Disambiguation with Constraint Propagation. *Proceedings of the 28th Meeting of the Association for Computational Linguistics (ACL)*, Pittsburgh, PA, pp. 31–38.

Menzel, W. and Schröder, I. (1998), Decision procedures for dependency parsing using graded constraints. *In Kahane, S. and Polguère, A. (eds), Proceedings of the Workshop on Processing of Dependency-Based Grammars*, pp. 78–87.

Nivre, J. (2005), Dependency Grammar and Dependency Parsing. *Technical Report MSI report 05133*, Växjö University: School of Mathematics and Systems Engineering.

Pennington, C-A. and McCoy, K-F. (1998), *Providing Intelligent Language Feedback for Augmentative Communication Users*, (Mittal, V-O., Yanco, H-A., Aronis, J. and Simpson, R. eds.), in Assistive Technology and Artificial Intelligence, Applications in Robotics, User Interfaces, and Natural Language Processing, LNAI 1458, Springer Verlag, 1998.

Peña-Casanova, J. (1991), *Normalidad, semiología y patología neuropsicológicas. Programa Integrado de Exploración Neuropsicológica. Test Barcelona*. Barcelona: Masson, 1991.

Reichle, J. and Drager, K. (2010), *Examining Issues of Aided Communication Display and Navigational Strategies for Young Children with Developmental Disabilities*, Journal of Developmental and Physical Disabilities, 2010, Vol. 22, Num. 3, pp. 289-311.

Sleator, D. and Temperley, D. (1993). Parsing English with a Link Grammar. *Third International Workshop on Parsing Technologies (IWPT)*, pp. 277–292.

Soro, E., Basil, C., Rosell, C. and Boix, J. (2007), *Vocabulari CACE-UTAC*, Grup d'Investigació per l'Estudi del Llenguatge i la CAA, Universitat de Barcelona. [http://www.utac.cat/noticies/utac-cacejadisponibleperadescarregarenquatreversions]

Takasaki, T. and Mori, Y. (2007), *Design and Development of a Pictogram Communication System for Children Around the World*, Lecture Notes in Computer Science, 2007, Vol. 4568, Intercultural Collaboration, pp. 193-206.

Taylor, A., Arnott, J., Alm, N. (2001), *Visualisation to Assist Non-speaking Users of Augmentative Communication Systems*. In: Proceedings of the International Conference of the IEEE on Information Visualisation, London, UK, pp. 251–256.

Tóth, B., Németh, G., Kiss, G. (2004), *Mobile Devices Converted into a Speaking Communication Aid*, In Proceedings of Computers Helping People with Special Needs, 9th ICCHP 2004, Paris, France, pp. 1016-1023.

Vaillant, P. (1997), *A Semantics-Based Communication System for Dysphasic Subjects*, Lecture Notes in Computer Science, Vol. 1211, Artificial Intelligence in Medicine, pp. 381-392.

Vanderheyden, P. and Pennington, C. (1998), *An augmentative communication interface based on conversational schemata*. Lecture Notes in Computer Science, Springer, Berlin, Vol. 1458, pp. 109-125.

Youngseo, K., Eunsil, L., Dohyun N., Taisung, H., Yoseob, W. and Hongki, M. (2006), *Verb Prediction and the Use of Particles for Generating Sentences in AAC System*, Lecture Notes in Computer Science, Volume 4304, AI 2006: Advances in Artificial Intelligence, pp. 997-1001.

# Annexes

## Appendix A: CACE vocabulary[59]

| | |
|---|---|
| ANIVERSARI | Activitats i festes |
| BON NADAL | Activitats i festes |
| BON ANY | Activitats i festes |
| CARNAVAL | Activitats i festes |
| CASAMENT | Activitats i festes |
| CASTANYADA | Activitats i festes |
| COLÒNIES | Activitats i festes |
| DEURES | Activitats i festes |
| ENCÀRREC | Activitats i festes |
| EXCURSIÓ | Activitats i festes |
| FESTA | Activitats i festes |
| FISIOTERÀPIA | Activitats i festes |
| GIMNÀSTICA | Activitats i festes |
| LA MONA | Activitats i festes |
| LOGOPÈDIA | Activitats i festes |
| LLENGUA | Activitats i festes |
| MATEMÀTIQUES | Activitats i festes |
| MÚSICA | Activitats i festes |
| NADAL | Activitats i festes |
| NATURALS | Activitats i festes |
| PARE NOËL | Activitats i festes |
| PASSAR LLISTA | Activitats i festes |
| PESSEBRE | Activitats i festes |
| PREMI | Activitats i festes |
| REGAL | Activitats i festes |
| REIS MAGS | Activitats i festes |
| SANT JOAN | Activitats i festes |
| SANT JORDI | Activitats i festes |
| SETMANA SANTA | Activitats i festes |
| SOCIALS | Activitats i festes |
| TIÓ | Activitats i festes |
| VACANCES | Activitats i festes |
| VIATGE | Activitats i festes |
| AIGUA | Aliments i begudes |
| ALIMENTS | Aliments i begudes |
| AMANIDA | Aliments i begudes |
| ARRÒS | Aliments i begudes |
| BEGUDES | Aliments i begudes |
| BERENAR | Aliments i begudes |
| CAFÈ | Aliments i begudes |
| CANELONS | Aliments i begudes |
| CARAMEL | Aliments i begudes |
| CARN | Aliments i begudes |
| CEREALS | Aliments i begudes |
| CIGRONS | Aliments i begudes |
| CIRERES | Aliments i begudes |
| COCA | Aliments i begudes |
| COCA-COLA | Aliments i begudes |
| COLA-CAO | Aliments i begudes |
| CRISPETES | Aliments i begudes |
| CROISSANT | Aliments i begudes |
| CROQUETES | Aliments i begudes |
| DÀTILS | Aliments i begudes |

[59] Here is a small sample of the vocabulary in CACE. Words in green are included in our system. The full vocabulary is on the digital version of appendix A.

| | |
|---|---|
| BARALLAR-SE | Verbs |
| BEURE | Verbs |
| CAMINAR | Verbs |
| CANTAR | Verbs |
| CAURE | Verbs |
| CANVIAR | Verbs |
| CANVIAR-SE | Verbs |
| CELEBRAR | Verbs |
| CLASSIFICAR | Verbs |
| COMPRAR | Verbs |
| CONCENTRAR-SE | Verbs |
| CONTAR | Verbs |
| CÓRRER | Verbs |
| COSIR | Verbs |
| CUINAR | Verbs |
| DESCANSAR | Verbs |
| DESPULLAR | Verbs |
| DIBUIXAR | Verbs |
| DONAR | Verbs |
| DORMIR | Verbs |
| EIXUGAR | Verbs |
| ENFILAR | Verbs |
| ENGANYAR | Verbs |
| ESCOLTAR | Verbs |
| ESCRIURE | Verbs |
| ESPERAR | Verbs |
| ESQUIAR | Verbs |
| ESTIMAR | Verbs |
| ESTIRAR-SE | Verbs |
| ESTRIPAR | Verbs |
| ESTUDIAR | Verbs |
| EXPLICAR | Verbs |
| FER | Verbs |
| FER CACA | Verbs |
| FER PETONS | Verbs |
| FER PIPI | Verbs |
| FOTOCOPIAR | Verbs |
| FUMAR | Verbs |
| GUANYAR | Verbs |
| GUARDAR | Verbs |
| INVITAR | Verbs |
| JUGAR | Verbs |
| LLEGIR | Verbs |
| MENJAR | Verbs |
| MIRAR | Verbs |
| MOLESTAR | Verbs |
| MOSTRAR | Verbs |
| MULLAR | Verbs |
| NECESSITAR | Verbs |
| NEDAR | Verbs |
| NETEJAR | Verbs |
| OBLIDAR | Verbs |
| OBRIR | Verbs |

## Appendix B: Verb patterns[60]

| verb | id | idverb | pronominal | pseudoimpe | tipusfrase | defaulttens | subj | subjdef | theme | themetipus | themedef |
|---|---|---|---|---|---|---|---|---|---|---|---|
| agafar | 1 | 1 | 0 | 0 | enunciativa | present | human | 1 | 1 | noun | això |
| agafar | 2 | 1 | 0 | 0 | peticio | present | human | 2 | opt | noun | |
| agradar | 3 | 2 | 0 | 1 | enunciativa | present | verb | una cosa | 0 | | |
| agradar | 4 | 2 | 0 | 1 | enunciativa | present | noun | una cosa | 0 | | |
| ajudar | 5 | 3 | 0 | 0 | imperatiu | present | human | 2 | opt | verb | |
| ajudar | 6 | 3 | 0 | 0 | imperatiu | present | human | 2 | opt | noun | |
| al·lucinar | 7 | 4 | 0 | 0 | enunciativa | present | human | 1 | opt | verb | |
| al·lucinar | 8 | 4 | 0 | 0 | enunciativa | present | human | 1 | opt | noun | |
| amagar | 9 | 5 | 0 | 0 | enunciativa | present | animate | 1 | opt | noun | |
| amagar-se | 10 | 6 | 1 | 0 | enunciativa | present | animate | 1 | 0 | | |
| anar | 11 | 7 | 0 | 0 | enunciativa | present | animate | 1 | 0 | | |
| anar | 12 | 7 | 0 | 0 | enunciativa | present | animate | 1 | 0 | | |
| aparcar | 13 | 8 | 0 | 0 | enunciativa | present | human | 1 | opt | vehicle | |
| baixar | 14 | 9 | 0 | 0 | enunciativa | present | animate | 1 | opt | lloc | |
| baixar | 15 | 9 | 0 | 0 | enunciativa | present | animate | 1 | opt | noun | |
| baixar | 16 | 9 | 0 | 0 | peticio | present | human | 2 | opt | noun | |
| ballar | 17 | 10 | 0 | 0 | enunciativa | present | human | 1 | opt | cançó (inanimat) | |
| ballar | 18 | 10 | 0 | 0 | peticio | present | human | 2 | opt | cançó (inanimat) | |
| banyar-se | 19 | 11 | 1 | 0 | enunciativa | present | animate | 1 | 0 | | |
| banyar-se | 20 | 11 | 1 | 0 | petició | present | human | 1 | 0 | | |
| barallar-se | 21 | 12 | 1 | 0 | enunciativa | present | animate | 1 | opt | noun | |
| beure | 22 | 13 | 0 | 0 | enunciativa | present | animate | 1 | opt | beguda | |
| beure | 23 | 13 | 0 | 0 | petició | present | human | 1 | 1 | beguda | alguna cosa |
| beure | 24 | 13 | 0 | 0 | petició | present | human | 2 | 1 | beguda | alguna cosa |
| caminar | 25 | 14 | 0 | 0 | enunciativa | present | human | 1 | 0 | | |
| cantar | 26 | 15 | 0 | 0 | enunciativa | present | animate | 1 | opt | cançó (inanimat) | |
| caure | 27 | 16 | 0 | 0 | enunciativa | present | noun | 1 | 0 | | |
| caure | 28 | 16 | 0 | 1 | enunciativa | present | noun | una cosa | 0 | | |
| canviar | 29 | 17 | 0 | 0 | enunciativa | present | human | 1 | opt | noun | |
| canviar-se | 30 | 18 | 1 | 0 | enunciativa | present | animate | 1 | 1 | noun | alguna cosa |
| canviar-se | 31 | 18 | 1 | 0 | petició | present | human | 1 | opt | noun | |
| canviar | 32 | 17 | 0 | 0 | enunciativa | present | human | 1 | opt | noun | |
| celebrar | 33 | 19 | 0 | 0 | enunciativa | present | human | 1 | 1 | noun | |

[60] This is a sample of the first 33 verb patters. The whole set of patterns is on the digital version of appendix B.

| verb | themeprep | receiver | receiverdef | receiverprep | benef | beneftipus | benefdef | benefprep | acomp | acompdef | acomprep |
|---|---|---|---|---|---|---|---|---|---|---|---|
| agafar | | 0 | | | opt | animate | | per | 0 | | |
| agafar | | 1 | mi | a | opt | animate | | per | 0 | | |
| agradar | | 1 | mi | a | 0 | | | | 0 | | |
| agradar | | 1 | mi | a | 0 | | | | 0 | | |
| ajudar | a | 1 | mi | a | 0 | | | | 0 | | |
| ajudar | amb | 1 | mi | a | 0 | | | | 0 | | |
| al·lucinar | amb | 0 | | | 0 | | | | 0 | | |
| al·lucinar | amb | 0 | | | 0 | | | | 0 | | |
| amagar | | opt | | a | 0 | | | | opt | | amb |
| amagar-se | | 0 | | | 0 | | | | opt | | amb |
| anar | | 0 | | | 0 | | | | opt | | amb |
| anar | | 0 | | | 0 | | | | opt | | amb |
| aparcar | | 0 | | | 0 | | | | opt | | amb |
| baixar | | 0 | | | 0 | | | | opt | | amb |
| baixar | | 0 | | | 0 | | | | opt | | amb |
| baixar | | 1 | mi | a | 0 | | | | opt | | amb |
| ballar | | 0 | | | 0 | | | | opt | | amb |
| ballar | | 0 | | | 0 | | | | opt | | amb |
| banyar-se | | 0 | | | 0 | | | | opt | | amb |
| banyar-se | | 0 | | | 0 | | | | opt | | amb |
| barallar-se | amb | 0 | | | opt | noun | | per | 0 | | |
| beure | | 0 | | | 0 | | | | opt | | amb |
| beure | | 0 | | | 0 | | | | opt | | amb |
| beure | | 0 | | | 0 | | | | opt | | amb |
| caminar | | 0 | | | 0 | | | | opt | | amb |
| cantar | | opt | | a | 0 | | | | opt | | amb |
| caure | | 0 | | | 0 | | | | 0 | | |
| caure | | 1 | mi | a | 0 | | | | 0 | | |
| canviar | | 0 | | | opt | noun | | per | opt | | amb |
| canviar-se | de | 0 | | | opt | noun | | per | opt | | amb |
| canviar-se | de | 0 | | | 0 | | | | 0 | | |
| canviar | | 0 | | | 0 | | | | opt | | amb |
| celebrar | | 0 | | | opt | noun | | per | opt | | amb |

| verb | tool | tooldef | toolprep | manera | maneradef | maneratipus | locto | loctotipus | loctodef | loctoprep |
|---|---|---|---|---|---|---|---|---|---|---|
| agafar | opt | | amb | opt | | adv | 0 | | | |
| agafar | opt | | amb | opt | | adv | 0 | | | |
| agradar | 0 | | | opt | | quant | 0 | | | |
| agradar | 0 | | | opt | | quant | 0 | | | |
| ajudar | 0 | | | 0 | | | 0 | | | |
| ajudar | 0 | | | 0 | | | 0 | | | |
| al·lucinar | 0 | | | opt | | adv | 0 | | | |
| al·lucinar | 0 | | | opt | | adv | 0 | | | |
| amagar | opt | | amb | opt | | adv | 0 | | | |
| amagar-se | opt | | amb | opt | | adv | 0 | | | |
| anar | opt | | amb | opt | | adv | 1 | noun | un lloc | a |
| anar | opt | | amb | opt | | adv | 1 | verb | fer una cosa | a |
| aparcar | 0 | | | opt | | adv | 0 | | | |
| baixar | opt | | amb | opt | | adv | opt | | | a |
| baixar | opt | | amb | opt | | adv | opt | | | a |
| baixar | opt | | amb | opt | | adv | opt | | | a |
| ballar | 0 | | | opt | | adv | 0 | | | |
| ballar | 0 | | | opt | | adv | 0 | | | |
| banyar-se | 0 | | | opt | | adv | 0 | | | |
| banyar-se | 0 | | | opt | | adv | 0 | | | |
| barallar-se | opt | | amb | opt | | adv | 0 | | | |
| beure | opt | | amb | opt | | adv | 0 | | | |
| beure | opt | | amb | opt | | adv | 0 | | | |
| beure | opt | | amb | opt | | adv | 0 | | | |
| caminar | opt | | amb | opt | | adv | opt | | | cap a |
| cantar | opt | | amb | opt | | adv | 0 | | | |
| caure | 0 | | | opt | | adv | 0 | | | |
| caure | 0 | | | 0 | | | 0 | | | |
| canviar | 0 | | | opt | | adv | 0 | | | |
| canviar-se | 0 | | | opt | | adv | 0 | | | |
| canviar-se | 0 | | | opt | | adv | 0 | | | |
| canviar | opt | | amb | opt | | adv | opt | | | a |
| celebrar | 0 | | | opt | | adv | 0 | | | |

| verb | locfrom | loctipus | locfromdef | locfromprep | locat | locatdef | locatprep | time | expressio | subverb |
|---|---|---|---|---|---|---|---|---|---|---|
| agafar | opt | | | de | opt | | a | opt | | 0 |
| agafar | opt | | | de | 0 | | | opt | si us plau | 0 |
| agradar | 0 | | | | 0 | | | opt | | 1 |
| agradar | 0 | | | | 0 | | | opt | | 0 |
| ajudar | 0 | | | | 0 | | | opt | si us plau | 1 |
| ajudar | 0 | | | | 0 | | | opt | si us plau | 0 |
| al·lucinar | 0 | | | | opt | | a | opt | | 1 |
| al·lucinar | 0 | | | | opt | | a | opt | | 0 |
| amagar | 0 | | | | opt | | a | opt | | 0 |
| amagar-se | 0 | | | | opt | | a | opt | | 0 |
| anar | 0 | | | | 0 | | | opt | | 0 |
| anar | 0 | | | | 0 | | | opt | | 0 |
| aparcar | 0 | | | | 1 | un lloc | a | opt | | 0 |
| baixar | opt | | | des de | 0 | | | opt | | 0 |
| baixar | opt | | | de | 0 | | | opt | | 0 |
| baixar | opt | | | de | 0 | | | opt | si us plau | 0 |
| ballar | 0 | | | | opt | | | opt | | 0 |
| ballar | 0 | | | | opt | | | opt | si us plau | 0 |
| banyar-se | 0 | | | | opt | | a | opt | | 0 |
| banyar-se | 0 | | | | opt | | a | opt | | 0 |
| barallar-se | 0 | | | | opt | | a | opt | | 0 |
| beure | 0 | | | | opt | | a | opt | | 0 |
| beure | 0 | | | | opt | | a | opt | | 0 |
| beure | 0 | | | | opt | | a | opt | | 0 |
| caminar | opt | | | des de | opt | | per | opt | | 0 |
| cantar | 0 | | | | opt | | a | opt | | 0 |
| caure | opt | | | de | opt | | a | opt | | 0 |
| caure | opt | | | de | opt | | a | opt | | 0 |
| canviar | 0 | | | | opt | | a | opt | | 0 |
| canviar-se | 0 | | | | opt | | a | opt | | 0 |
| canviar-se | 0 | | | | opt | | a | opt | si us plau | 0 |
| canviar | opt | | | de | 0 | | | opt | | 0 |
| celebrar | 0 | | | | opt | | a | opt | | 0 |

## Appendix C: Annotated lexicon[61]

| id | imgnom | nomtext | mf | singpl | contabincontab | determinat | plural | femeni |
|---|---|---|---|---|---|---|---|---|
| 340 | NULL | jo | masc | sing | contable | sense | jo | NULL |
| 341 | NULL | ell | masc | sing | contable | sense | ells | NULL |
| 342 | NULL | ella | fem | sing | contable | sense | elles | NULL |
| 343 | NULL | nosaltres | masc | pl | contable | sense | nosaltres | NULL |
| 344 | NULL | vosaltres | masc | pl | contable | sense | vosaltres | NULL |
| 345 | NULL | ells | masc | pl | contable | sense | ells | elles |
| 346 | NULL | logopeda | masc | sing | contable | 0 | logopedes | logopeda |
| 347 | NULL | mare | fem | sing | contable | 1 | mares | NULL |
| 348 | NULL | mestre | masc | sing | contable | 0 | mestres | mestra |
| 349 | NULL | metge | masc | sing | contable | 0 | metges | metgessa |
| 350 | NULL | monitor | masc | sing | contable | 0 | monitors | monitora |
| 351 | NULL | monstre | masc | sing | contable | 0 | monstres | NULL |
| 352 | NULL | nen | masc | sing | contable | 0 | nens | NULL |
| 353 | NULL | nena | fem | sing | contable | 0 | nenes | NULL |
| 354 | NULL | noi | masc | sing | contable | 0 | nois | NULL |
| 355 | NULL | noia | fem | sing | contable | 0 | noies | NULL |
| 356 | NULL | nòvio | masc | sing | contable | 1 | nòvios | nòvia |
| 357 | NULL | tiet | masc | sing | contable | 1 | tiets | tieta |
| 358 | NULL | pare | masc | sing | contable | 1 | pares | NULL |
| 359 | NULL | perruquer | masc | sing | contable | 0 | perruquers | perruquera |
| 360 | NULL | persona | fem | sing | contable | 0 | persones | NULL |
| 361 | NULL | policia | masc | sing | contable | 0 | policíes | policia |
| 362 | NULL | tu | masc | sing | contable | sense | tu | NULL |
| 363 | NULL | veí | masc | sing | contable | 1 | veïns | veïna |
| 364 | NULL | mal | masc | sing | contable | sense | mals | NULL |
| 365 | NULL | por | fem | sing | incontable | sense | pors | NULL |
| 366 | NULL | calor | fem | sing | incontable | sense | calor | NULL |
| 367 | NULL | fred | masc | sing | incontable | sense | fred | NULL |
| 368 | NULL | paciència | fem | sing | incontable | sense | paciència | NULL |
| 369 | NULL | sorpresa | fem | sing | contable | 0 | sorpreses | NULL |
| 370 | NULL | abric | masc | sing | contable | 0 | abrics | NULL |
| 371 | NULL | arrecades | fem | pl | contable | 0 | arrecades | NULL |
| 372 | NULL | banyador | masc | sing | contable | 0 | banyadors | NULL |

[61] This is a fragment of the table with the annotated nouns in the system. All the tables for all the annotated words in the system are in the digital version of appendix C.

## Appendix D: Developed code[62] & Appendix E: Full code within framework

```php
function generateSentence($patternfinal, $propietatsfrase, $partpreguntaposada)
{
    $pattern = new Mypattern();
    $pattern = $patternfinal;

    // Indiquem que si el temps per defecte és l'imperatiu, que la frase és una ordre a no ser que estigui activat el modificador
    // de desig o permís que tenen preferència o que hi hagi una partícula de pregunta.
    if ($propietatsfrase['tense'] == "defecte" && $pattern->defaulttense == "imperatiu" && (!$propietatsfrase['tipusfrase'] == "desig"
            || !$propietatsfrase['tipusfrase'] == "permis" || !$partpreguntaposada)) {
        $propietatsfrase['tipusfrase'] = "ordre";
    }
    else if ($partpreguntaposada) $propietatsfrase['tipusfrase'] = "pregunta";

    // 1. Ordenem els slots segons el tipus de frase
    $pattern->ordenarSlotsFrase($propietatsfrase);

    // 2 i 3. 2: Ordenar paraules de dins dels slots, ja posant les preposicions. 3: Controlar que les paraules concordin en gènere i
    // número (els adjs amb els noms i la PartPregunta "quant" amb el theme, si hi és). Afegir també les coordinacions només de NOMS,
    // ADJECTIUS i ADVERBIS DE MANERA
    $pattern->ordenarSlotsInternament();

    // 4. Posar articles als noms

    $pattern->putArticlesToNouns($propietatsfrase["tipusfrase"]);

    // 5. Conjugar els verbs

    $pattern->conjugarVerbs($propietatsfrase);

    // 6. Treure els "jo" i "tu" dels subjectes. Canviar receivers a pronoms febles i posar-los a darrere el verb si cal. Posar
    // modificadors de frase com el "no" o el "també". Fusionar preposicions amb articles (de+el/s = del/s... a+el, per+el...).
    // Posar apòstrofs de preps i pronoms febles (i guions?). Netejar espais abans o després dels apòstrofs.
    // Escriure la frase final, posant les expressions i altres advs de temps al final.

    $pattern->launchCleaner($propietatsfrase["tipusfrase"]);

    return $pattern->printFraseFinal();
}
```

[62] The part of code above is the function of the generator that calls each of the generator's modules.
The code developed by us during the project is separated in folders in digital appendix D. In total there are over 8500 lines of code distributed in 16 files.
The full code within the framework CodeIgniter is also separated in folders in digital appendix E.

## Appendix F: Apostrophising rules of singular definite articles in Catalan

As a general rule, the definite articles "el" and "la" are reduced to "l'" when the word that follows begins with a vowel or a silent "h": "l'avi", "l'invent", "l'euga", "l'ull", l'hivern", etc. This apostrophising rule presents the following exceptions, in addition to the common apostrophising rules of the definite articles and the preposition "de" exposed later on:

1. Before a feminine word beginning with an unstressed "i" or "u" (preceded or not by a silent "h"), the full form of the article "la" is maintained: "la idea", "la il·lusió", "la hipòtesi", "la història", "la unitat", "la universitat", "la humitat", etc.
2. In front of the words "una" (when it refers to an hour), "ira" and "host", the full form is used: "la una", "la ira", "la host".

Remarks: In front of Roman and Arabic numerals, the article adopts the reduced form only if when written in alphabetic characters the article is also apostrophied: "l'1 de gener" (= "l'u de gener), l'XI (= "l'onze), etc.

**Common apostrophising rules of the definite articles and the preposition "de":**

1. When the following word begins with a non vocalic "i" or "u", the full form of the article, both masculine and feminine, is maintained: "el iaio", "el ioga", "el iogurt", "el hiatus", "el hioide", "el uigur", "el uombat", "la iarda", "la ionosfera", "la hialita", "la hiena", etc. Exception: "l'ió".
2. Definite articles are not apostrophised in front of borrowed words or foreign proper nouns that start with a voiced "h": "el hippy", etc.
3. Definite articles are never apostrophised before the name of the letters of the alphabet: "la a", "la ene", "la hac", "la ele", etc.
4. The common rules for apostrophising are also applied to words in quotation marks or italics. The only exception is when italics are used for metalinguistic purposes.
5. Preceding numerical symbols or abbreviations, which start with a vocalic sound, definite articles are also apostrophised: "l'1 de maig", "l'XI Congrés", "l'ap. 4", etc.
6. General rules are also applied to acronyms, which are read like a word: "l'ONU", "l'IVA", "l'URSS", "la UNESCO", "la UEFA", etc.
7. Definite articles are apostrophised before acronyms, which are spelled and which start with a vocalic sound: "l'EMT", "l'AVL", "l'ONG", "l'FM", "l'LSD", etc.

8. In front of borrowed or foreign words that begin with a liquid "s", the masculine definite article accepts apostrophe or lack of it ("el speaker" or "l'speaker"), while the feminine definite always presents its full form ("la Scala"). Nevertheless, the most coherent solution is to never apostrophise in any case: "el speaker", "la Scala", etc.

## Appendix G: Set of 100 test sentences

| | | |
|---|---|---|
| **1** | He menjat molt. | I ate a lot. |
| **2** | Vull anar al lavabo. | I want to go to the bathroom. |
| **3** | Quina hora és? | What time is it? |
| **4** | Anirem al restaurant. | We will go to the restaurant. |
| **5** | El vestit és nou. | The dress is new. |
| **6** | Teniu tomàquets? | Have you got tomatoes? |
| **7** | Vaig caure. | I fell. |
| **8** | El meu gos és molt graciós. | My dog is very funny. |
| **9** | Puc jugar a pilota, si us plau? | Can I play with the ball, please? |
| **10** | Està marejat. | He/She is sick. |
| **11** | On és la meva nina? | Where is my doll? |
| **12** | No tinc la cadira de rodes. | I don't have the wheelchair. |
| **13** | M'agrada la neu. | I like snow. |
| **14** | Vull dormir. | I want to sleep. |
| **15** | Els macarrons són molt bons. | Macarroni are very good. |
| **16** | A la tarda necessitaré el medicament. | In the afternoon, I will need the medicine. |
| **17** | Dona'm una forquilla, si us plau. | Give me the fork, please. |
| **18** | El monitor no és a l'escola. | The monitor is not at the school. |
| **19** | Vull una poma. | I want an apple. |
| **20** | La piscina és molt guai. | The swimming pool is very cool. |
| **21** | Tot és molt divertit. | Everything is very fun. |
| **22** | Vindré tard. | I will come late. |
| **23** | No vull més verdura. | I don't want any more vegetables. |
| **24** | Quantes croquetes vols? | How many croquettes do you want? |
| **25** | Tinc fred. | I'm cold. |
| **26** | Vols que anem a casa? | Do you want us to go home? |
| **27** | Espera'm al bar. | Wait for me at the bar. |
| **28** | Vaig a la biblioteca. | I go to the library. |
| **29** | Amb qui vas? | Who are you going with? |
| **30** | Ahir, vaig comprar unes sabates blaves. | Yesterday, I bought a pair of blue shoes. |
| **31** | Un bitxo molt raro s'ha amagat darrere la porta. | A very weird bug hid behind the door. |
| **32** | És un pal. | It's a bum. |
| **33** | No vinguis. | Don't come. |
| **34** | Tinc molta por. | I'm really afraid. |
| **35** | Vine. | Come. |
| **36** | Quan vindreu a casa? | When are you coming home? |

| | | |
|---|---|---|
| **37** | Vindrem demà. | I'll come tomorrow. |
| **38** | Menjo un gelat de xocolata. | I eat a chocolate ice-cream. |
| **39** | M'agrada llegir llibres de por. | I like reading terror books. |
| **40** | Avui no agafis l'abric. | Today, do not grab the coat. |
| **41** | A l'estiu ens banyarem a la platja. | In summer we will bath in the sea. |
| **42** | Ahir vaig anar al metge. | Yesterday I went to the doctor. |
| **43** | L'home del jersei negre és dolent. | The man on the black jersey is mean. |
| **44** | No m'agrada el pastís de poma. | I don't like apple pie. |
| **45** | Dimecres anirem a cantar. | We will sing on Wednesday. |
| **46** | Hem begut amb el got trencat. | We drank with the broken glass. |
| **47** | Què fa la mare? | What is mother doing? |
| **48** | La infermera és molt alegre. | The nurse is very cheerful. |
| **49** | Vull classificar-me per les olimpíades. | I want to classify for the Olympics. |
| **50** | Un iogurt, si us plau. | A yoghurt, please. |
| **51** | Parla molt? | He/She speaks a lot. |
| **52** | La meva germana llegeix malament. | My sister reads poorly. |
| **53** | M'he classificat primer. | I got first place. |
| **54** | Ho sento. | I'm sorry. |
| **55** | Hola! | Hello! |
| **56** | El sol és taronja i vermell a la tarda. | The sun is orange and red in the afternoon. |
| **57** | Un got d'aigua, si us plau. | A glass of water, please. |
| **58** | Estic bé. | I'm fine. |
| **59** | Felicitats! | Congratulations! |
| **60** | Ajuda'm a baixar les escales, si us plau. | Help me go down the stairs, please. |
| **61** | Estàvem molt contents. | We were very happy. |
| **62** | Espera un minut. | Wait a minute. |
| **63** | No vull que et barallis. | I don't want you to fight. |
| **64** | El tren anava molt lent. | The train was going very slowly. |
| **65** | Això serà una sorpresa. | This will be a surprise. |
| **66** | Tinc mal al peu. | My foot hurts. |
| **67** | Les cinc. | Five o'clock. |
| **68** | T'estimo. | I love you. |
| **69** | El cuiner i la cuinera fan pastissos. | The cook(male) and the cook (female) make cakes. |
| **70** | Tu ets molt més alt. | You are much taller. |
| **71** | Per què? | Why? |
| **72** | A qui ho explico? | To whom I explain it? |
| **73** | Dijous serem a Europa. | We will be in Europe on Thursday. |

| | | |
|---|---|---|
| **74** | Com ho sabrem? | How are we going to know? |
| **75** | La mare està amb el pare. | Mother is with father. |
| **76** | No. | No. |
| **77** | No ho sé. | I don't know. |
| **78** | Canvia la cadira de color marró. | Change the brown chair. |
| **79** | Ja està. | I'm done. / It's done. |
| **80** | Mala sort. | Too bad. |
| **81** | Espero que vinguis demà. | I hope that you'll come tomorrow. |
| **82** | Amaga't. | Hide yourself. |
| **83** | Celebren el seu aniversari al parc. | They celebrate his birthday at the park. |
| **84** | Compra això a la farmàcia. | Buy this at the pharmacy. |
| **85** | Vam beure cafè al casament. | We drank coffee at the wedding. |
| **86** | Cinc persones estaven incòmodes dins el taxi. | Five people were uncomfortable in the taxi. |
| **87** | La meva gossa necessita una dutxa. | My dog needs a shower. |
| **88** | Horrible. | Horrible. |
| **89** | Bon any. | Happy new year. |
| **90** | La meva amiga està trista. | My friend is sad. |
| **91** | He aparcat el cotxe a la plaça. | I parked the car at the square. |
| **92** | Cent euros. | A hundred euros. |
| **93** | La meva germana descansa sobre el llit. | My sister rests on the bed. |
| **94** | Explica'm un conte. | Tell me a story. |
| **95** | Després mirarem una pel·lícula al cine. | Afterwards, we will watch a movie at the cinema. |
| **96** | Jugareu més tard. | You will play later. |
| **97** | El bebè caminarà aviat. | The baby will walk soon. |
| **98** | També vull una bufanda lila i llarga. | I also want a long purple scarf. |
| **99** | Bona nit. | Good night. |
| **100** | Adéu. | Bye bye. |

## Appendix H: Raw input, results and scores from the annotators[63]

| idusu | tipusfrase | negat | tense | frase | inputwords | parses | gene | generatorstring |
|---|---|---|---|---|---|---|---|---|
| 2 | enunciativa | 0 | perfet | 1 | menjar / molt | 10 | 9 | He menjat molt. |
| 5 | enunciativa | 0 | perfet | 1 | jo / menjar / molt | 9 | 9 | He menjat molt. |
| 10 | enunciativa | 0 | perfet | 1 | jo / menjar / molt | 10 | 9 | He menjat molt. |
| 2 | enunciativa | 0 | defecte | 10 | ell / estar / marejat | 10 | 9 | Ell està marejat. |
| 5 | enunciativa | 0 | defecte | 10 | ell / estar / marejat | 10 | 5 | Ell està marejat. |
| 10 | enunciativa | 0 | defecte | 10 | ell / estar / marejat | 10 | 9 | Ell està marejat. |
| 10 | enunciativa | 0 | defecte | 100 | adéu | 10 | 9 | Adéu. |
| 5 | enunciativa | 0 | defecte | 100 | adéu | 10 | 9 | Adéu. |
| 2 | enunciativa | 0 | defecte | 100 | adéu | 10 | 9 | Adéu. |
| 2 | enunciativa | 0 | defecte | 11 | on / ser / meu / nina | 10 | 9 | On és la meva nina? |
| 5 | pregunta | 0 | defecte | 11 | on / ser / meu / nina | 10 | 9 | On és la meva nina? |
| 10 | enunciativa | 0 | defecte | 11 | on / ser / meu / nina | 10 | 9 | On és la meva nina? |
| 2 | enunciativa | 1 | defecte | 12 | tenir / cadira de rodes | 10 | 9 | No tinc la cadira de rodes. |
| 5 | enunciativa | 1 | defecte | 12 | jo / tenir / cadira de rodes | 10 | 9 | No tinc la cadira de rodes. |
| 10 | enunciativa | 1 | defecte | 12 | tenir / cadira de rodes | 10 | 9 | No tinc la cadira de rodes. |
| 2 | enunciativa | 0 | defecte | 13 | agradar / neu | 10 | 9 | La neu m'agrada. |
| 5 | enunciativa | 0 | defecte | 13 | jo / agradar / neu | 10 | 8 | La neu m'agrada. |
| 10 | enunciativa | 0 | defecte | 13 | jo / agradar / neu | 10 | 9 | La neu m'agrada. |
| 2 | desig | 0 | defecte | 14 | dormir | 10 | 9 | Vull dormir. |
| 5 | desig | 0 | defecte | 14 | dormir | 10 | 9 | Vull dormir. |
| 10 | desig | 0 | defecte | 14 | dormir | 10 | 9 | Vull dormir. |
| 2 | enunciativa | 0 | defecte | 15 | macarrons / ser / molt / bo | 10 | 9 | Els macarrons són molt bons. |
| 5 | enunciativa | 0 | defecte | 15 | macarrons / ser / molt / bo | 10 | 9 | Els macarrons són molt bons. |
| 10 | enunciativa | 0 | defecte | 15 | macarrons / ser / molt / bo | 10 | 9 | Els macarrons són molt bons. |
| 2 | enunciativa | 0 | futur | 16 | necessitar / medicament / tarda | 10 | 9 | Necessitaré el medicament a la tarda. |
| 10 | enunciativa | 0 | futur | 16 | tarda / jo / necessitar / medicament | 10 | 9 | Necessitaré el medicament a la tarda. |
| 5 | enunciativa | 0 | futur | 16 | tarda / jo / necessitar / medicament | 10 | 8 | Necessitaré el medicament a la tarda. |
| 2 | ordre | 0 | defecte | 17 | donar / forquilla / si us plau | 10 | 9 | Dóna'm una forquilla, si us plau. |
| 10 | ordre | 0 | defecte | 17 | tu / donar / forquilla / si us plau | 10 | 9 | Dóna'm una forquilla, si us plau. |
| 5 | ordre | 0 | defecte | 17 | tu / donar / forquilla / jo / si us plau | 10 | 9 | Dóna'm una forquilla, si us plau. |
| 2 | enunciativa | 1 | defecte | 18 | monitor / ser / escola | 10 | 9 | El monitor no és a l'escola. |
| 10 | enunciativa | 1 | defecte | 18 | monitor / ser / escola | 10 | 9 | El monitor no és a l'escola. |
| 5 | enunciativa | 1 | defecte | 18 | monitor / ser / escola | 10 | 9 | El monitor no és a l'escola. |
| 2 | desig | 0 | defecte | 19 | poma | 10 | 9 | Vull una poma. |
| 10 | enunciativa | 0 | defecte | 19 | jo / voler / poma | 10 | 9 | Vull una poma. |
| 5 | desig | 0 | defecte | 19 | poma | 10 | 9 | Vull una poma. |
| 2 | desig | 0 | defecte | 2 | anar / lavabo | 10 | 9 | Vull anar al lavabo. |
| 5 | desig | 0 | defecte | 2 | jo / anar / lavabo | 10 | 9 | Vull anar al lavabo. |
| 10 | desig | 0 | defecte | 2 | jo / anar / lavabo | 10 | 9 | Vull anar al lavabo. |
| 2 | enunciativa | 0 | defecte | 20 | piscina / ser / molt / guai | 10 | 9 | La piscina és molt guai. |
| 10 | enunciativa | 0 | defecte | 20 | piscina / ser / molt / guai | 10 | 9 | La piscina és molt guai. |
| 5 | enunciativa | 0 | defecte | 20 | piscina / ser / molt / guai | 10 | 9 | La piscina és molt guai. |
| 2 | enunciativa | 0 | defecte | 21 | tot / ser / molt / divertit | 10 | 9 | Tot és molt divertit. |
| 5 | enunciativa | 0 | defecte | 21 | tot / ser / molt / divertit | 10 | 9 | Tot és molt divertit. |
| 10 | enunciativa | 0 | defecte | 21 | tot / ser / molt / divertit | 10 | 9 | Tot és molt divertit. |
| 2 | enunciativa | 0 | futur | 22 | venir / tard | 10 | 9 | Vindré tard. |
| 5 | enunciativa | 0 | futur | 22 | venir / tard | 10 | 9 | Vindré tard. |

[63] Above there is a small fragment of the inputs, results and scores given by the annotators to the 100 test sentences. In green are sentences that all annotators input equally, in yellow sentences that one annotator input differently and in red sentences that all annotators input differently. The rightmost column has the generated sentence by the *compansion* system. Some columns have been removed for better visibility. The rest of the document is in the digital version of appendix H.

## Appendix J: Detailed input results by sentences

| | |
|---|---|
| # 3 annotators equally | 10, 100, 14, 15, 18, 20, 21, 27, 3, 31, 4, 43, 45, 46, 5, 54, 55, 56, 59, 6, 61, 64, 69, 70, 71, 75, 77, 79, 80, 83, 84, 85, 86, 89, 90, 93, 94, 96, 97, 99 |
| # 2 annotators equally | 1, 11, 12, 13, 16, 19, 2, 22, 25, 26, 28, 32, 33, 34, 36, 35, 38, 39, 40, 41, 42, 44, 47, 48, 49, 50, 51, 52, 53, 58, 60, 62, 63, 65, 66, 67, 68, 7, 72, 73, 76, 78, 8, 82, 87, 88, 9, 91, 92, 95 |
| # 3 annotators differently | 17, 23, 24, 29, 30, 37, 57, 74, 81, 98 |
| | |
| # Omit subject | 1, 12, 13, 16(/), 17(/), 19(/), 2, 22(/), 23(/), 24(/), 25, 26, 28, 29(/), 30(/), 33(/), 34, 36(/), 35, 38(/), 39, 40(/), 42, 44, 49(/), 50(/), 53, 58, 60, 62(/), 63, 66, 7, 78, 82, 9, 91, 98(/) |
| # Omit receiver | 17(/) |
| # Order of the possessives | 52, 8, 87 |
| # Order of the quantifiers | 32, 38(/), 48, 57(/), 62(/), 65 |
| # Different sentence modifiers used | 11, 19(/), 22(/), 23(/), 24(/), 29(/), 30(/), 33(/), 36(/), 37(/), 40(/), 47, 49(/), 50(/), 57(/), 67, 74(/), 76, 81(/), 88, 92, 95, 98(/) |
| # Order of the time expressions | 16(/), 37(/), 41, 73, 81(/) |
| # Other words' order | 51, 68, 72, 74(/) |
| | |
| # Good when equally (perfect or understandable) | 10, 100, 14, 15, 18, 20, 21, 27, 3, 31, 4, 43, 45, 46, 5, 54, 55, 56, 59, 6, 61, 64, 69, 70, 71, 75, 77, 79, 80, 83, 84, 85, 86, 89, 90, 93, 94, 96, 97, 99 |
| # Bad when equally | |
| | |
| # Good when differently (perfect or understandable) | 1, 11, 12, 13, 16, 17, 19, 2, 22, 23, 24, 25, 26, 28, 29, 30, 32, 33, 34, 36, 37, 35, 38, 40, 41, 42, 44, 47, 48, 50, 51, 52, 53, 39(/), 49(/), 57(/), 58, 60, 62, 63, 65, 66, 67, 68, 7, 72, 73, 74, 76, 78, 8, 81, 82, 87, 88, 9, 91, 92, 95, 98 |
| # Bad when differently | 39(/), 49(/), 57(/) |
| # Identical output when differently | 1, 11, 12, 13, 16, 17, 19, 2, 22, 23, 24, 25, 28, 29, 30, 32, 33, 34, 36, 37, 35, 38, 40, 41, 42, 44, 47, 48, 49, 51, 52, 53, 58, 60, 62, 63, 65, 66, 67, 68, 7, 72, 73, 74, 76, 78, 8, 82, 87, 88, 9, 91, 92, 95, 98 |

## Appendix K & L: Scores of the parser and of the generator sentence by sentence[64]

| | 10 | 7 | 3 | 0 |
|---|---|---|---|---|
| **1** | 3 | 0 | 0 | 0 |
| **2** | 3 | 0 | 0 | 0 |
| **3** | 3 | 0 | 0 | 0 |
| **4** | 2 | 1 | 0 | 0 |
| **5** | 3 | 0 | 0 | 0 |
| **6** | 1 | 2 | 0 | 0 |
| **7** | 3 | 0 | 0 | 0 |
| **8** | 3 | 0 | 0 | 0 |
| **9** | 3 | 0 | 0 | 0 |
| **10** | 2 | 1 | 0 | 0 |
| **11** | 3 | 0 | 0 | 0 |
| **12** | 3 | 0 | 0 | 0 |
| **13** | 2 | 1 | 0 | 0 |
| **14** | 3 | 0 | 0 | 0 |
| **15** | 3 | 0 | 0 | 0 |
| **16** | 2 | 1 | 0 | 0 |
| **17** | 3 | 0 | 0 | 0 |
| **18** | 3 | 0 | 0 | 0 |
| **19** | 3 | 0 | 0 | 0 |
| **20** | 3 | 0 | 0 | 0 |
| **21** | 3 | 0 | 0 | 0 |
| **22** | 3 | 0 | 0 | 0 |
| **23** | 3 | 0 | 0 | 0 |
| **24** | 3 | 0 | 0 | 0 |
| **25** | 3 | 0 | 0 | 0 |
| **26** | 3 | 0 | 0 | 0 |
| **27** | 3 | 0 | 0 | 0 |
| **28** | 3 | 0 | 0 | 0 |
| **29** | 0 | 3 | 0 | 0 |
| **30** | 3 | 0 | 0 | 0 |
| **31** | 3 | 0 | 0 | 0 |
| **32** | 2 | 1 | 0 | 0 |
| **33** | 3 | 0 | 0 | 0 |
| **34** | 3 | 0 | 0 | 0 |
| **35** | 3 | 0 | 0 | 0 |
| **36** | 2 | 1 | 0 | 0 |
| **37** | 2 | 1 | 0 | 0 |
| **38** | 3 | 0 | 0 | 0 |
| **39** | 0 | 1 | 2 | 0 |
| **40** | 3 | 0 | 0 | 0 |

[64] Above, there is a sample of the scores of the first 40 sentences given to the generator by the annotators. The full version of this table was the one used to calculate the Kappa of the generator. The whole table of scores for the generator and the parser are in appendix L and K respectively. The appendixes also include the values for Kappa and other data. Both of them are in different spreadsheets of the same Excel file.

## Appendix M: Detailed generator results by sentences

### Category 2: Subcategories

| | | | |
|---|---|---|---|
| # There are minor errors in the word order of the generated sentence. | 13, 16, 37, 41, 44, 45 | 6/26 | 23,10% |
| # There are minor errors in the choice of the articles (mainly definite vs indefinite) of the generated sentence. | 6, 39, 40, 46, 69, 78, 91 | 7/26 | 26,92% |
| # There are minor errors with the conjugation of the verbs. | 61, 81, 86 | 3/26 | 11,54% |
| # There are other minor errors, which are not described in a-c. | 4, 10, 29, 32, 36, 49, 51, 60, 64 | 9/26 | 34,62% |
| # There are minor errors that come from the parsing. | 66 | 1/26 | 3,85% |

### Category 4: Subcategories

| | |
|---|---|
| # The sentence is wrongly generated and cannot be understood, although the parsing was correct. | 49 |
| # The sentence is wrongly generated and cannot be understood and the parsing was also incorrect. | |
| # The sentence was not generated at all, due to an internal error of the system. | |