A Calendar Interface in French: XIPAgenda

Claude Roux
Xerox Research Centre Europe
6, chemin de Maupertuis, 38240 Meylan, France
Claude.roux@xrce.xerox.com
+33 4 76 61 51 38

ABSTRACT
In this paper we describe a French language interface to a calendar system. The system has been successfully implemented using state-of-the art technologies in parsing. We show how temporal expressions are analyzed and how this interface simplifies the task of setting your agenda.

Author Keywords
NLP, French, calendar, parser, XIP, temporal expression.

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION
Most companies rely today on calendar software to schedule meetings and events. The use of these agendas has become so ubiquitous that most people take them for granted. However, setting a new event, even though most GUI are pretty simple to use, has become a bit of a tedious task as the number of options has steadily but incredibly increased over the last years. Of course, adding a new meeting to your agenda does not require a full week training. It takes less than a minute for most users to fill in the necessary template to add a meeting or a rendezvous. However, a current calendar system involves some clicking and some typing that may prove cumbersome. In a typical software from a well-known company from Seattle, between seven to ten different manipulation are needed in order to correctly insert one single appointment. The user must first select the day, then the hour, then split his/her message into a title, a location, a list of attendees, set the correct duration. Again, this list of tasks is neither complex nor difficult, but since the calendar template over the years has gained in options what it has lost in clarity, it definitely requires some careful writing. It is a well known problem that mouse manipulations are usually slower than keyboard strokes, as using a mouse requires moving a hand off the keyboard, which stops the writing flow. The most efficient way would be to speak to your computer, using a voice recognition system, to automatically update your agenda. Unfortunately, the state of voice recognition technology is far from offering such a feat. Furthermore, the use of voice recognition software in an open space does pose some etiquette problems, which have not been solved up to now. If we cannot talk to our computer, we can still write to it.

The goal of this article is to describe how a natural language interface with a broad coverage French grammar can improve the communication of a human being with his/her machine. The principal difficulty in this task is the parsing and the understanding of the temporal expression which has been typed in by the user. As most experiences in the domain of NLP interfaces have shown in the past, the patience of users with a system that would not analyze most of her/his propositions usually thins very quickly. We will describe our system: *XIPAgenda*, which implements a natural language interface to a calendar. *XIPAgenda* is used both for instantiating and querying the agenda.

AN AGENDA ITEM?
XIPAgenda purpose is to allow a user, with no particular training, to be able to add a meeting in a calendar, using only her/his knowledge of her/his own language. However, before describing in more details our system, it would be interesting to study what a calendar item looks like.

An agenda item can be modeled as a function: $\Psi(\theta_0,\theta_1,\tau_0,\tau_1,\delta_0,\delta_1,\lambda,\rho,\mu)$, where:

1. θ_0,θ_1 are initial and final dates.
2. τ_0,τ_1 are initial and final times
3. δ_0, δ_1 is first the time lag from a referent time and the duration of the event.
4. λ is a location
5. ρ is a list of persons (*attendees*)
6. μ is the topic of the event

The goal of our system is to map a sentence over this representation.

Unfortunately, while you may find in a template a specific slot for each of these data, a sentence rarely split into any of these categories easily. Most of these data might be either absent or worse implicit.

Example
The meeting on technology transfer will take place in 20mn, starting at 10:00AM and will last half an hour, in the Everest room.

If we analyze this sentence, we can compute the following dates and times. First, the referent date will be today, as

none is provided in the sentence. We will suppose that θ_0,θ_1 are then seeded with today's date. Second, since, no exact time has been provided τ_0,τ_1 will be initialized with the time the message was written. Third, we have two durations, which have been provided. The first one *20mn* maps over δ_0 and will be used to compute the exact starting date of the meeting. The second duration *"half an hour"* is mapped over δ_1 and will be used to compute the exact date of the end of the meeting. The location λ is "*Everest room*". The list of persons ρ is empty and the topic μ will be restricted to "*meeting on technology transfer"*, since no other information is available. As we can see on this simple example, the definition of a simple meeting can prove quite complex to process.

Temporal Expression

Frank Schilder [1] gives a list of different sorts of temporal expressions that one may find in a text:

- Explicit, *the exact date is provided*
- Indexical: *tomorrow, yesterday*
- Duration: *for two hours*
- Vague: *in a few days*

An appointment statement might fall in each of these categories. However, we will discard the *vague* one as we hope users to set precise appointments.

At first glance, the limitation of our system to only simple agenda statements, one sentence at time, would sound as too simple a task. It would be reasonable to think that extracting temporal expressions from texts would not require very complex grammars. Yet, in evaluation campaigns such as TREC or MUC, the existence of *when* questions such as:

- *When did Napoleon die?*
- *When was President Reagan first elected?*

have shown that this job was far from being trivial. It has triggered a renewed interest in this task and a variety of new systems to detect and solve these expressions has emerged. Different approaches exist. For instance, since machine learning techniques have become prevalent in the last years in the NLP domains, some serious attempts have been made to build temporal corpora such as TIMEBANK (see [2]). This corpus has been annotated with TimeML (see [4]), which is an XML annotation schema, based on TIMEX2. TIMEX2 (see [5]) provides a descriptive language to make some specific temporal data more explicit. In other words, the goal is to transform a natural language time description into a tractable item. The goal is to translate the time descriptions into actual dates.

Example

The following sentence has been flagged according to the TIMEX2 annotation schema:

In January, the weather was very mild.

In <**TIMEX VAL= "2007-01-XX-X"**>*January*</TIMEX>, *the weather was very mild.*

The TimeML annotation schema is much more ambitious than TIMEX2 as it also provides a structure in which events are annotated and linked together through temporal expressions. The TIMEBANK has been used to train machine learning algorithms as in [7] approach, in which their system is trained to *temporally order* events in documents. However, the use of machine learning techniques to build temporal expression extractors would be rather difficult to use in an agenda application. First, the training requires hundred of sentences, already annotated, second the correction of wrong analyses usually imposes the annotation of more sentences with a re-training of corpus, with limited confidence that the new model is effectively corrected. Unfortunately, there is no available corpus in French to train a system on temporal expression, which seriously hampers the use of these methods in our case. Another interesting system, which has been developed for the German language, is COSMA (see [8]). COSMA is a language server, which analyzes the content of e-mails and detects any agenda items. It then tries to set an appointment that would be suitable to all participants. The system is based on a full-fledged grammar, which is composed of rules. The use of a symbolic grammar is both lighter in terms of computing power (statistical models tend to be quite greedy when running) and simpler to correct. This assumption is especially true when the goal is to detect a subset of natural language expressions, which can be described *in extenso*. Yet, we should keep in mind that a limited set of potential expressions does not translate into a simple list of regular expressions. For instance, Google Calendar provides a tool which accepts agenda description in plain English. The analysis of the sentences is made with a set of regular expressions that are extremely sensitive to variations, and often fail to provide the correct output. We have used the Xerox Incremental Parser (e.g. XIP) with an updated grammar for the French language, in order to meet both our requirements in terms of coverage and efficiency.

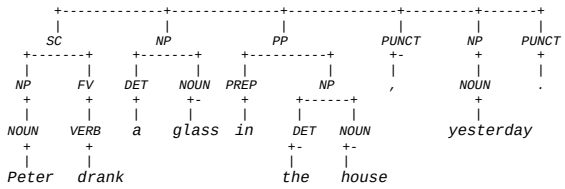
THE PARSING ENGINE: XIP

The Xerox Incremental Parser (e.g. XIP), (see [9,10]) is a symbolic parser which has been developed at the Xerox Research Centre Europe (e.g. XRCE) in the last years. This parser provides a broad coverage grammar for at least seven languages. The output of XIP is composed of a syntactic node tree (also called chunk tree*)* and syntactic functions (or dependencies). A syntactic function may link together two or more chunk nodes, or simply defines the nature of one single node.

Example

Peter drank a glass in the house, yesterday.

The output of the English grammar is the following chunk tree:



And a list of syntactic functions:

- SUBJECT(*drink,Peter*)
- OBJECT(*drink,glass*)
- PERSON(*Peter*)
- LOCATION(*house*)
- DATE(*yesterday*)

XIP extracts the different dates, locations and person names which are present in a text. It also extracts some important syntactic relations such a SUBJECT and OBJECT. The example here is in English, but a sentence in French would yield similar syntactic functions. The XIP engine is not new to temporal extraction or question/answering techniques and it has been successfully used in different campaigns such as EQUER (see [11]) or TempEval (see [12]). Thanks to these campaigns, we have at our disposal a large grammar for French, with some specific set of rules dedicated to temporal extractions. Furthermore, a large amount of work has been dedicated to the definition of specific sub-grammars embedded within regular grammars to add name entity detection for languages such English and French (see [14]). Thus, the analysis of a sentence as below:

- *Pierre Dupont s'est rendu à Paris en 2001 (Pierre Dupont traveled to Paris in 2001)*

already yields the following list of dependencies:

- PERSON(*Pierre Dupont*)
- LOCATION(*Paris*)
- DATE(*2001*)

TEMPORAL GRAMMAR

Yet, this list of functions is insufficient to map any sentences to an agenda. The DATE function lacks many features which would help our system to translate them into exact dates. Furthermore, we also need to detect times and durations, which would be important to any appointment definitions. Let's examine the following sentences:

- Meeting in two hours *time lag*
- Meeting at 2:00PM *exact time*
- 30mn meeting in one hour *duration and time lag*

A study of these sentences shows how versatile the description of meetings can be.

The French grammar has been modified in order to enrich it with new functions that are mapped over our agenda item function: $\Psi(\theta_0,\theta_1,\tau_0,\tau_1,\delta_0,\delta_1,\lambda,\rho,\mu)$.

The French grammar, which has been re-designed for our task, yields now the following dependencies: *DATE, TIME, DURATION, LOCATION, PERSON, TOPIC.* Furthermore, each of these functions can be enriched with a list of specific features that gives some more details.

Features

A date, for instance might have many different formats:

- *the 15th*
- *April, the 3rd*
- *2007-09-05*
- *On Monday*
- *Tomorrow*

Each of these dates requires a different set of features in order to refine their description. These features are the following: *YEAR, MONTH, MDIGIT, DDIGIT, DAY, SHIFT.* The difference between *MONTH* and *MDIGIT* is simply that in one case, the month is a word, while in the second case it is a number. The analysis of the above strings gives the following results:

- DATE_DDIGIT(*15th*)
- DATE_MONTH_DDIGIT(*April, the 3rd*)
- DATE_YEAR_MDIGIT_DDIGIT(*2007-09-05*)
- DATE_DAY(*Monday*)
- DATE_SHIFT(*tomorrow*)

The grammar can also set two more features: *START* and *END* which corresponds to the initial and the final dates in the sentence. These two features are also associated to the *TIME* function in the same fashion.

Example

- *Meeting tomorrow from 10:00AM to 11:00AM, in room 20.*

The analysis of the above sentence will give the following functions:

- DATE_SHIFT(*tomorrow*)
- TIME_START(*10:00AM*)
- TIME_END(*11:00AM*)
- LOCATION(*room 20*)

A similar French agenda description would give the following analysis:

- *Réunion de deux heures, lundi à 13h00, en salle 12. (Two hours meeting, Monday at 1 :00PM, in room 12)*

The resulting functions are:

- DATE_START_DAY(*lundi*)
- DURATION (*two hours*)
- TIME_START(*13 h00*)
- LOCATION(*salle 12*)

As we can see on this example, the grammar has taken some decisions about the *START* feature added to the functions: *DATE* and *TIME*. The strategy goes as follow: *when only one date or one time is found in a sentence, then we add the feature* START *to their corresponding functions.*

Time Calculus

The grammar does not compute any actual dates; this job is passed to a specific piece of program, which in our architecture has been written in Python. The role of the grammar is to detect linguistic occurrences of date and time in a sentence, not to compute any actual values. However, thanks to the different features that the grammar adds to the different functions, the definition of a date is quite simple. Each feature leads to a specific sub-case of date computing, which makes the process quite straightforward. For instance, on a *DATE_DAY(Monday)* instance, the program will jump to the *day* resolution part of our program to determine the exact date of that *Monday*. Missing elements from our agenda function Ψ, will be replaced with today's values, or will be computed on the fly. The different sorts of duration, time start or time end will then be mixed together to compute the correct values. The purpose of this piece of program is to calculate two date representations, whose format will be compliant with our agenda software. Usually, a complete date format on a computer will involve year, month, day, hour, minute and second values, which will be merged into one single string. The program is left with the task of filling in the blank: the main implicit datum is that the referent date is today.

Location and Persons

The location extraction is part of the generic French grammar. It is automatically detected in a statement and a LOCATION function is yielded, which the calling program will use to set the location field in the agenda template. We have a similar behavior for the PERSON function, which is also part of the regular French grammar. The values of these functions will be passed to our program, which will set the corresponding fields in our agenda application.

TOPIC

The goal of an agenda is to inform a user of an imminent appointment on time. Usually, a small window pops up, which displays a few lines to describe the next appointments. However, the only data available to define μ in our Ψ function is the initial statement. In a preliminary prototype, the subject line was the statement itself, a solution that was far from being satisfactory. The title was both too long and too redundant with the agenda itself. The other solution is to remove all redundant information that is already known to the system, such as dates, times and the locations.

Example

- Réunion, le 15 novembre à 18h00, en salle Mont-Blanc, pour proposer une offre à Metal Corp. *(Meeting on November the 15th at 6 :00PM, in Mont-blanc room to propose an offer to Metal Corp.)*

The objective is to remove from the above sentence all the words that would be related to locations or temporal expressions, in order to store the following line:

- Réunion pour proposer une offre à Metal Corp (*Meeting to propose an offer to Metal Corp)*

This operation is very similar to summarization techniques such as the sentence compression method used in [13]. The sentence compression consists in removing sub-clauses and other less essential information from a sentence in order to produce a shorter sentence, which still retains most of its original meaning. The operation consists in deleting specific phrases (or chunks) instead of removing independent words. A chunk such as *à 18h00* has a certain autonomy within a sentence, which a word such as *18h00* does not have. This autonomy means that removing one sole word will unbalance the sentence, while removing the chunk will have a certain impact on the sentence meaning without destroying its readability.

Example

If we compare the two sentences below:

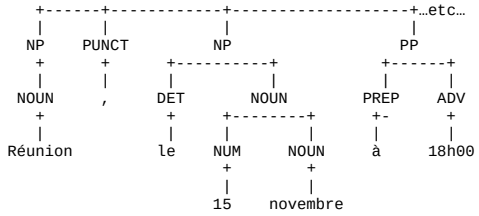
- **Réunion, le 15 novembre* ***à****, en salle Mont-Blanc, pour proposer une offre à Metal Corp.*
- *Réunion, le 15 novembre, en salle Mont-Blanc, pour proposer une offre à Metal Corp.*

In the first sentence, the time *18h00* was removed, while leaving the preposition *à.* This sentence is ungrammatical. The second sentence, where the full chunk was removed is still a valid French sentence.

Chunk Tree

The use of a parser is absolutely central to this task. In XIP, since the building of the chunk tree is preliminary to all others operation, we have enough material to proceed to the compression. Once, the syntactic functions have been computed, the nodes involved in temporal expressions are marked with a specific feature.

Example
The chunk tree of our previous sentence is the following:



We have only represented on this tree the temporal nodes that we want to remove.

The grammar has also yielded the following functions:

- DATE_MONTH_DDIGIT(*15 novembre*)
- TIME(*18h00)*

The compression in this case consists in marking the nodes that are involved in these temporal expressions, together with their top nodes: *NP* and *PP*. Once all these nodes have been discovered and isolated, they can be safely removed from the sentence. This method allows us to generate a short but informative subject line.

QUERIES

Most calendars provide a search engine, which is usually limited to only keyword search. This search possibility is sometimes enhanced with some categories, which can be used to sort out appointments according to their content, such as *business, birthday, meeting etc.* This list is usually restricted to a few domains and cannot be easily expanded. Furthermore, a message can only be categorized with one category at a time.

Ontologies

The only way to give a message some background is to enrich it with more information so that the search engine will be able to find similar items. This enrichment is done through an ontological description of what a calendar action is. For instance, a *meeting* in French can be described in many different ways:

- It could be another noun such as: *réunion* or *rendez-vous*.
- It could be a verb such as *voir (see), discuter (discuss), parler (speak).*
- It could also be an expression such as: *Donner une presentation (give a presentation)* or *Organiser un événement (to organize an event).*

XIPAGENDA has been enriched with an ontological description of most of these words. Whenever, an appointment statement is processed, the system automatically analyzed each word and returns a list of related words. The calling program uses this list to build a *body* in which hyperonyms and hyponyms of these words are automatically appended to the initial statement. This enlarged *body* is then stored together with the other information into the calendar. Thus a simple sentence such as: *Meeting with Peter tomorrow,* is automatically enriched with the following elements: *reunion, appointment.* When a user looks for a given appointment, it can search for *meeting* or *appointment* even though these words were not present in the message in the first place. This enrichment allows a limited calendar application to become *semantically aware*, despite the fact that it only provides a simple search interface.

Querying

Querying a calendar is very similar to defining a statement, to one exception: The dates in this case could be in the past. Furthermore, while the duration in most agenda statements is within a day, a query might cover a full month or even a year.

Example
Below is a list of questions about past statements:

- Réunion en janvier? *(meeting in January)*
- Présentation en 2007? *(meeting in June)*
- Conférence la semaine dernière ? *(conference last week)*

In a first version of our prototype, once the initial and final dates had been computed, we would build a query in which the keywords and these dates were grouped into one single instruction that was sent to the calendar application. However, this method proved very slow and was replaced by a complete download of all agenda items for a given period of time into a python dictionary, which was then indexed according to different strings computed on:

1. Year
2. Year/Month
3. Year/Month/Day
4. Year/Month/Day/Hour
5. Week number
6. Words from the statements
7. Statement strings

The reason behind this architecture is that querying for a period of time or for an exact date corresponds to one single access to that list. For instance, a query on *last week*, would automatically return the list of appointments recorded for "*current week -1"*. The list was quite tractable as the number of agenda items for one single person rarely exceeds a few hundred lines a year.

Querying this structure with a sentence is done in a four steps:

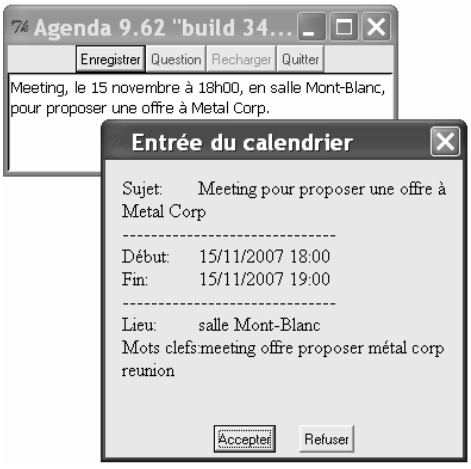
1. The sentence is parsed
2. The temporal information is computed
3. The system searches our python dictionary for each word from the statement, for each location and for each temporal data. Each element in this dictionary is a list of appointments.
4. Only elements common to all lists are returned as a result.

Example
A query such as:

- Conférence la semaine dernière ? *(conference last week)*

Corresponds to a search on : *"week number"* and *conference.*

ARCHITECTURE
The *XIPAGENDA* is written in python, using the basic *Tkinter* graphical library to handle windows. XIP provides both an internal and external python API. The internal API allows a rule, anywhere in the grammar, to call a python procedure, while the external API transforms XIP into a python library that can be freely imported into a python program. Below is an example of the window which is used to enter an appointment. The user might press *enregistrer (record)* to add a new item in his agenda or he might press *Question* to query the calendar application. If the statement ends with a question mark: "?", the system will automatically consider the statement as a query, even though *enregistrer* is clicked. A small window pops up with a summary of the information that was extracted from the statement by the grammar. The user might then decide to store this item to his/her calendar application.



The summary proposes a *subject* line, which is a compression of our initial statement. The two other lines are the beginning and the end of the appointment. The *lieu* is the location of the meeting. The *Mots Clefs* (keywords) section is a list of all potential important words from the statement together with some synonyms.

We have linked our application to *Outlook*© through a COM interface, but our solution could easily be adapted to other applications as the python program first builds a specific calendar object that is then mapped over the COM *Outlook*© interface.

EVALUATION
The evaluation was made on 104 different sentences which were mainly extracted from e-mails. Below is a sample of some of these sentences:

- Voyage du 15 au 18 mars (*Travel from the 15th to the 18th)*
- Dîner, le 2 à 19h00, dans l'auberge Napoléon, 7 rue Montorge (*Dinner, the 2nd at 7 :00PM, in the Auberge Napoléon, 7 rue Montorge)*
- Retour à Grenoble, le 23/02/2007 à 19h38 (*return to Grenoble, the 23rd at 19 :38)*
- Conférence téléphonique, demain à 9h00, dans le bureau de Laurent, pour le projet ALPHA (*Phone conference, tomorrow at 9 :00AM, in Laurent office, for the ALPHA project*)
- Réunion en Mont Blanc dans deux heures (*Meeting in Mont-Blanc in two hours)*
- Départ de Bruxelles le 15 février 2007 à 18h00, arrivée à Grenoble le 16 à 20h00. (*Departure from Bruxelles, February the 15th 2007 at 6 :00PM, arrival at Grenoble the 16th at 8 :00PM)*
- Conférence dimanche en huit en salle 12 *(Conference, Sunday after next Sunday, in room 12)*

The grammar had been designed over a different set of sentences. The issue in testing the system was that errors could stem from different parts of the system. The grammar could miss some elements from the sentence, or the python calling program could wrongly compute some specific dates. The definition of recall and precision was a real issue. Our preliminary idea was to compute precision and recall, sentence by sentence. If precision could be defined as the number of sentences that had a correct summary, we had more trouble to define a recall, as all sentences did receive a parse, even though it proved wrong in some cases.

We decided to compute our precision and recall using a different approach. We manually counted the number of potential dates, durations and locations that the whole corpus contained and we checked how many dates, times, durations and locations were actually extracted from our corpus by our system.

- The recall was defined as the number of dates, times, durations and locations extracted by

XIPAGENDA over the corpus versus the number that we had manually extracted.

- The precision was the number of correct dates, times, locations and durations that was computed by *XIPAGENDA*.

Our corpus contained 104 sentences with 259 actual definitions:

- 98 dates
- 81 times
- 25 durations
- 55 locations

The system then computed the exact initial and final dates for each of these definitions using both dates, times and durations. The recall and precision was computed on these pairs, together with the location data.

The recall was very high, which was a proof of the broad coverage of the grammar: *95%*

The precision was also very high, as the system did not try to cope with anything but specific temporal and location information in the different sentences: *94%*.

However, the number of sentences that were correct was only 85%. We rejected any sentences that would contain one or more errors.

We did not try to extract proper names in our experimentation, as our calendar system was not designed to use them. We decided to simply discard them for latter trials.

The system failed on certain date format. In French, the tradition is to use a pattern such as: DD/MM/YY, however, certain people use other formats such as: YYYY/MM/DD, which XIPAGENDA could not properly analyze.

The system also failed on sentences such as:

- En déplacement, les 14, 15, 16 et 17 septembre (*Away on September the 14th,15th, 16th and 17th)*

In the above sentence, the grammar could not cope with a list of days.

Finally, we also rejected one sentence whose *subject* line had been wrongly computed, due to an unexpected lexicon entry. *Pierre*, which is a very common first name in French, had been recorded in the lexicon with an unanticipated temporal feature, which eventually modified the full analysis of the statement: the word *Pierre* is used in the French expression: *l'Age de Pierre*, which means *Stone Age*.

- Voir Pierre, demain à 15h00 (*see Pierre, tomorrow at 3 :00PM)*

The final stage of the analysis failed to compute the exact date, which *Pierre* was supposed to be, while correctly computing the initial and final dates.

It should be noted that the grammar has been amended since and covers now almost 98% of all these sentences.

CONCLUSION

We have tried with *XIPAGENDA* to design a system which would both robust and utilizable. The robustness is a real challenge as each new user has her/his own way to describe a calendar event. However, our grammar is now quite reliable and thanks to the relative closure of this domain, covers most of the temporal expressions currently used in French. Our system proposes some specific features such as the possibility of searching the calendar with natural language queries, the compression of the sentence to build a comprehensible subject line and finally the integration of ontologies to enlarge the query coverage. Even though *XIPAGENDA* still presents some glitches, it has proven that the use natural language in such an application was feasible. The possibility to tell your computer to do things is an old dream, which when it works, even on such a small subset, is still enthralling. Most products today consist of an all-in-one application where e-mails, calendar, to-do list, and contacts are merged into one single program. However, even though this integration is usually well made, the use of these tools requires jumping from one window to another tab. Adding a contact or sending an e-mail involve filling in templates that have become more and more complex. Again, these applications are far from being excruciatingly complex and most people use them seamlessly. *XIPAGENDA* in our vision is only one step further to the complete control of these tools in natural language. We think that a NLP central could be designed which would allow people to add new contacts with a simple sentence, leaving the task of finding who is who and where (s)he lives to the computer. E-mails and to-do lists could also be managed with simple statements that would be recognized by the machine. Furthermore, NLP can also apply to the content of messages or e-mails. With such a NLP central, it becomes possible to spot dates and addresses in documents, and automatically proposes meeting or new contacts to a user. When such an application is put together, all textual information originating either from the user's commands or from external mails is linked into one single net of information, whose querying might provide in-sight on one's work which would be difficult otherwise.

REFERENCE

[1] Frank Schilder, Extracting meaning from temporal nouns and temporal prepositions, *ACM Transactions on Asian Language Information Processing (TALIP)*, (2004), 33-50.

[2] Pustejovsky, J., P. Hanks, R. Saurı, A. See, R. Gaizauskas, A. Setzer, D. Radev, B. Sundheim, D.Day, L. Ferro and M. Lazo. 2003b. The TIMEBANK Corpus. *Proceedings of Corpus Linguistics (2003)*: 647-656.

[3] Pustejovsky, J., R. Saurı, J. Castano, D. R. Radev, R. Gaizauskas, A. Setzer, B. Sundheim and G. Katz., Representing Temporal and Event Knowledge for QA Systems. Mark T. Maybury (ed.) *New Directions in Question Answering*. MIT Press, Cambridge, (2004).

[4] Pustejovsky, J., Saurı, R., Setzer, A., Gaizauskas, R., and Ingria, B., TimeML Annotation Guidelines. http://www.cs.brandeis.edu/~jamesp/arda/time/documentation/AnnotationGuideline-v0.4.0.pdf, (2002).

[5] Mani, I., Ferro, L., Sundheim, B., and Wilson, G., Guidelines for Annotating Temporal Information. *In Proceedings of the Human Language Technology Conference*, 2001.

[6] Dawson, F. Stenerson, D.. Internet Calendaring and Scheduling Core Object Specification (iCalendar), RFC2445, Internet Society, (1998)

[7] Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. *Machine Learning of Temporal Relations. Proceedings of ACL'2006*, (2006)

[8] S. Busemann, T. Decleck, A. K. Diagne, L. Dini, J. Klein, and S. Schmeier. Natural Language Dialogue Service for Appointment Scheduling Agents. *Proceedings of the Fifth Conference on Applied Natural Language Processing*, 1997, 25-32.

[9] Claude Roux. Phrase-driven parser. In *Proceedings of VEXTAL'99*, Venezia, Italia. VEXTAL'99, (1999)

[10] Salah Aıt-Mokhtar, Jean-Pierre Chanod and Claude Roux. Robustness beyond shallowness: incremental dependency parsing. *Special issue of the NLE Journal*, (2002).

[11] B. Grau, A.-L. Ligozat, I. Robba, A. Vilnat, and L. Monceaux. Frasques: A question-answering system in the EQUER evaluation campaign. *In LREC 2006, Genoa, Italia*, May 2006.

[12] Caroline Hagège and Xavier Tannier, XIP temporal module for TempEval campaign, Proceedings of SemEval workshop at ACL 2007 Prague, Czech Republic, (2007), p. 492-495

[13] Vandeghinste, Vincent and Yi Pan. Sentence compression for automated subtitling: A hybrid approach. In *Proceedings of the ACL Workshop on Text Summarization*. Barcelona, Spain, (2004), 89-95.

[14] Caroline Brun, Caroline Hagège, Intertwining Deep Syntactic Processing and Named Entity Detection, *LECTURE NOTES IN COMPUTER SCIENCE, Springer, (2004).*