Content Finder AssistanT

Conference Paper · February 2015

DOI: 10.1109/ICIN.2015.7073836

1 author:

Romain Laroche
Microsoft Maluuba
58 PUBLICATIONS **185** CITATIONS
SEE PROFILE

Content Finder AssistanT

Romain Laroche
Orange Labs
Issy les Moulineaux
France
Email: romain.laroche@orange.com

***Abstract*—This paper focuses on "how to design and develop a dialogue system with a minimal effort". It presents a novel model for automatic generation of dialogue systems built from contents. This approach is similar to and relies on a search engine, but with augmented dialogue capabilities: at each dialogue turn, the system propose n keywords, in order to optimise the information gain expectation. Its implementation, CFAsT, endeavours to keep the best from both worlds: the universality and automatic generation from search engines, and the usability, the assistance and the self optimisation provided by the dialogue systems. Thus, a beta dialogue application can be generated with no effort. It may serve to gather a dialogue corpus, or to gain a first return of experience. It can also be used as a low cost service. Afterwards, it can be improved with dedicated dialogue strategies.**

Keywords*—*Dialogue systems, Information retrieval

I. Introduction

Research on dialogue systems often focuses on "how to design and develop a system with a minimal effort". It appears that models that made their way to the industry are the ones that can be represented, shared, built, modified, and maintained easily. The big winners are indeed the automata-based dialogue systems [14], partly because it is a friendly way to display the dialogue flow [1], [30], and thus to conceive. The fully data-driven approach [15], [31], based on reinforcement learning [27], was presented as the next generation of dialogue systems (automatic design and optimisation). Over the years, this approach knew a lot of improvement, both in reinforcement learning techniques [29], [22], and in methodology [3], [6]. And yet, reinforcement learning has only been applied in industrial spoken dialogue systems when it was marginally projected into conventional systems [28], [23], [21], and many reservations expressed by the industrial branch of dialogue research [19], [20] are still topical.

This article proposes a novel data-driven model for dialogue systems that alleviates most of those reservations (see Section VII). Indeed, we propose to learn the dialogue strategy from contents, instead of doing so from the interaction with users. This behaviour constitutes a baseline which can be overridden with handcrafted rules, and/or online learnt strategies.

In this paper, we consider an information retrieval task, that is achieved through dialogue led from the contents themselves. The so-built dialogue systems can easily be compared to search engines [2], [16], and the model is actually built up on a search engine. But a content-based dialogue system and conventional search engines have different goals: search engines look for the best contents regarding a query while a dialogue system has the secondary task to help the user to formulate more accurately her/his query, accordingly to the content base. Query auto completion [18] and suggestions [8] are very rudimentary [25], as they are unable to recommend words that are several words distant despite being strongly associated with the query.

We aim to go one step further by analysing contents and by proposing a set of keywords that partition in an optimal way the relevant contents given a user query. It also involves the user in a dialogue that incites her/him to give a feedback on the dialogue success. This information is further used as an evaluation for optimising the underlying algorithms online. Moreover, the automatically generated application can easily be enhanced with handcrafted dialogue branches, disambiguation strategies and reusable behaviour on *outside perimeter* queries such as chatting about weather forecast or asking for a joke.

This model has been implemented as Content-Finder AssistanT (CFAsT). Currently, CFAsT is limited to text-based dialogue applications. It constitutes a first step, useful on web portals. It already goes further than performing question answering like Watson [7], but our ultimate goal is to generate low-cost spoken dialogue systems automatically that can be easily upgraded with applicative development.

The dialogue model is presented in Section II. The indexation and information retrieval are exposed in Section III. Then, Section IV describes the theory behind the keywords selection, which constitutes the main novelty of this article. Afterwards, we discuss the computational complexity of our algorithms in Section V and find a solution to deal with large databases of documents. Next, our implementation is described in section VI. And finally, Section VII discusses the results and the next steps.

II. Dialogue Model

A. Main dialogue flow

The basic dialogue flow is rather simple and is sketched in Figure 1. The dialogue begins at the black circle location. First, query and the list of forbidden contents (`fdbCont`) are set to empty. Then, it relies on a loop over a search engine returning the number of full match (result information) and the five best contents (result list). Then, the user has the choice between selecting one of the contents or to carry on refining her/his query. If a content is selected, it is (dis)played to the user and a question about the usefulness of the content is asked. This evaluation enables to learn from experience that have been validated by the user (see Subsection III-C). If the content is rejected, it is added to the forbidden content list and it will not be proposed again without restarting the dialogue from the beginning. Alternatively, there are two ways of refining

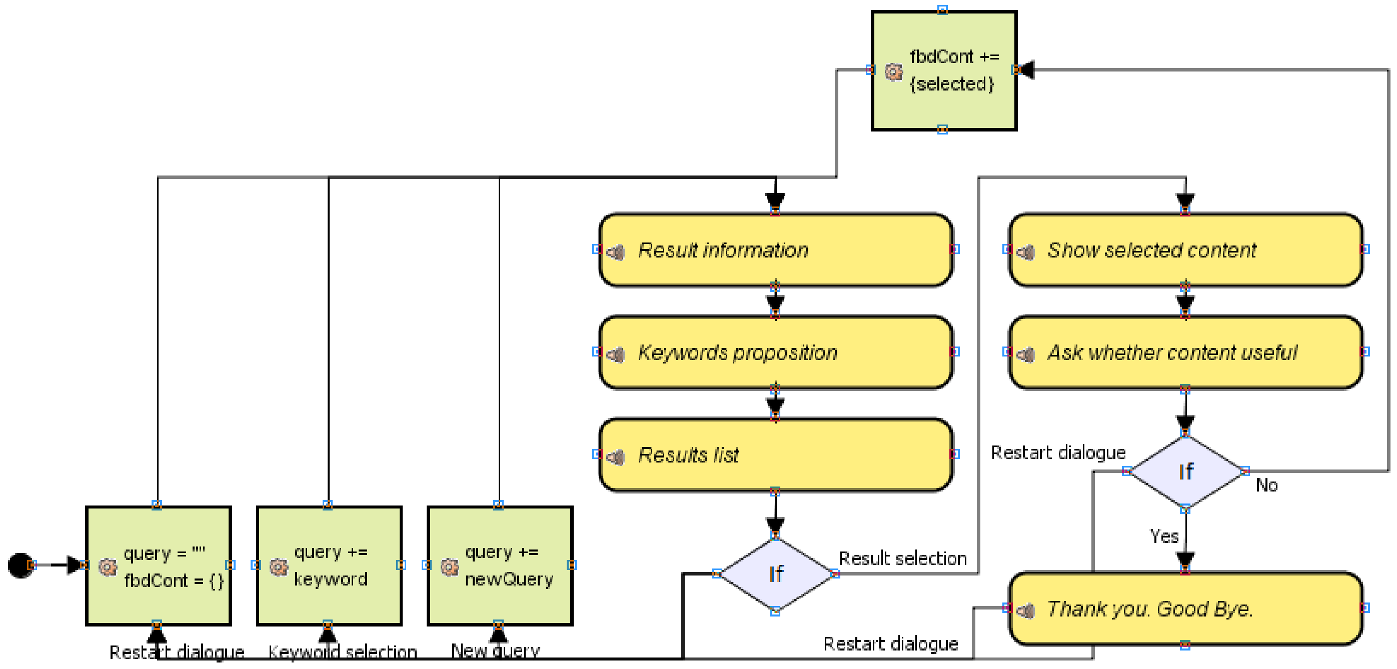


Fig. 1: Basic call flow in the content-based dialogue model.

the query: free input (voice or keyboard) or keyword selection (mouse or touch). The purpose of such a simple structure is to offer a beta version of any content-finder assistant at no cost.

B. *Ways to manually improve the generated system*

In addition to the dialogue flow, the generated system can be improved with expert knowledge by different means:

1) by using template-based dialogue rules overriding the default search engine strategy of the system:
   - to present directly a result if the query matches exactly the title of a content,
   - to clarify spotted ambiguities,
2) by designing full dialogue branches/scripts[1],
3) by adding domain knowledge with stop words and synonyms,
4) by overriding the keyword selection for given requests

III. Indexation and Information Retrieval

A. *Indexation*

In order to work efficiently, the search engine and the keyword selection need a pre-processing. The same pre-processing is applied to both contents and user queries. It comprises two steps: text pre-processing and numerical pre-processing.

A stemming algorithm is applied in order to avoid to consider different inflections as distinct words. As well, a stop word dictionary is used in order to remove all the words that do not convey meaning such as articles, prepositions or auxiliary/modal verbs. In the end of the text pre-processing, the output is a list of stems, also called keywords or terms in this paper.

The text pre-processing is followed with a numerical pre-processing in order to assign a weight to each term (keyword) for a given document (content). The weight $w_{t,d}$ of each term t in a document d is assessed thanks to the tf-idf formula with logarithmically scaled augmented frequency [9]:

$$w_{t,d} = atf(t,d) \times idf(t,\mathcal{D}) \tag{1a}$$

$$atf(t,d) = \frac{logf(t,d)}{\max_{t' \in d} logf(t',d)} \tag{1b}$$

$$logf(t,d) = \ln(1 + f(t,d)) \tag{1c}$$

$$idf(t,\mathcal{D}) = \ln \frac{|\mathcal{D}|}{|d \in \mathcal{D} : t \in d|} \tag{1d}$$

Where $f(t,d)$ is the frequency (number of occurrences) of term t in document d, $logf(t,d)$ is the logarithmic frequency, $idf(t,\mathcal{D})$ is the inverse document frequency of t inside the whole document database $\mathcal{D}$, $atf(t,d)$ is the augmented term frequency, $w_{t,d}$ is the weight of t in d, and $|X|$ is the cardinality of set X. For keyword selection (see Subsection IV), we also compute a relevance coefficient $\mathbb{1}_{t,d}$ in order to denote the probability that t is a relevant term in document d. It is a thresholded linear function of $w_{t,d}$ parametered with a constant α:

$$\mathbb{1}_{t,d} = \min(1, \alpha w_{t,d}) \tag{2}$$

[1]Contrarily to the others which are based on templates, this free way to improve the system requires software development.

B. Information Retrieval

The information retrieval relies on a Vector Space Model [24] and the relevance to a query q of any document d is given by the cosine similarity. But in our case, the normalisation by $\sqrt{\sum_t w_{t,d}}\sqrt{\sum_t w_{t,q}}$ is abandoned because its first part is already included in the augmented term frequency, and its second part is constant over the documents. As a consequence, the similarity function follows the following parametric Euclidean dot product formula:

$$\mu(q,d) = \sum_{t \in q} \xi_t w_{t,d} w_{t,q} \tag{3}$$

Where $\xi = \{\xi_t > 0\}_{t \in \mathcal{T}}$ is a weighting parameters associated to each term t, and $\mathcal{T}$ is the set of all terms encountered at least once in $\mathcal{D}$.

C. ξ optimisation

To train ξ, the method is to consider that, given a user query, a click on a document among n presented to the user, and the information that the document has been useful, it means that the clicked document d^* is more relevant than the $n-1$ others:

$$\mu(q,d^*) - \mu(q,d) \geq \epsilon \tag{4a}$$

$$\sum_{t\ \in q} \xi_t w_{t,q}(w_{t,d^*} - w_{t,d}) \geq \epsilon \tag{4b}$$

Then, we use a Support Vector Machine [4] to define ξ to satisfy those inequalities.

IV. Keyword selection algorithm

A. General Intuition

The novelty of this paper consists in creating automatically a dialogue structure from the data. Since we are limited by the capability to automatically generate a well-formed natural language question, the method we followed consists in proposing the most discriminative terms given the current request and the document basis.

We first took into consideration a co-clustering [5] of documents and terms. The problem we faced with this approach is that a cluster cannot be described by a single term. Actually, we do not need to cluster documents and we want to avoid to cluster terms, since we want to use a single keyword by option, in order to avoid an information overload [12].

We use the information theory [26] to compute the expected information gain after proposing a set of terms to the user, which can be reformulated as the document base centric entropy loss.

B. General formula

Let $\mathcal{T}$ be the set of terms encountered at least once in the documents. $\mathcal{T}^{-q}$ is this set minus the the terms of current request q[2]. The proposed algorithm scans the set of the terms subsets of cardinality n and maximise the answer utility expectation. To measure this, we use the information gain expectation, also called entropy in information theory. This is formalised as follows:

$$T_n^* = \underset{T \subset \mathcal{T}^{-q} : |T|=n}{\operatorname{argmax}} \ \mathcal{H}_T^q \tag{5a}$$

$$\mathcal{H}_T^q = \sum_{t \in T} p_T^q(t) \mathcal{I}^q(t) \tag{5b}$$

Where $\mathcal{H}_T^q$ reads out entropy over terms in T given query q ; where $p_T^q(t)$ is the probability that term t is selected when the set of terms T is proposed ; and where $\mathcal{I}^q(t)$ is the information gain after the selection of term t. Note that $\mathcal{I}^q(t)$ is independent of the other terms in T.

C. Computation of $p_T^q(t)$

To compute $p_T^q(t)$, we use the current document distribution according to request. To do that, we use the weight $\mu(q,d)$ computed with the similarity Equation 3. The following formula shows that $p_T^q(t)$ is the weighted average of the probability $p_T(t,d)$ for the user to select the term t if (s)he is looking for the document d:

$$p_T^q(t) = \frac{\sum_{d \in \mathcal{D}} \mu(q,d) p_T(t,d)}{\sum_{d \in \mathcal{D}} \mu(q,d)} \tag{6}$$

The probability $p_T(t,d)$ is computed from the probability that term t is relevant in the document d: $\mathbb{1}_{t,d}$. But several terms might be completely relevant for a given document. We need to compute for each possible outcome, how many terms are relevant, and which ones. To do so, we start with the computation of the probability that none of them are relevant:

$$p_T(\emptyset, d) = \prod_{t \in T} (1 - \mathbb{1}_{t,d}) \tag{7}$$

The probability $p_T(t,d)$ to select a term t, given that document d is targeted is therefore the probability that at least one term is relevant, times the probability to choose t among the relevant terms:

$$p_T(t,d) = (1 - p_T(\emptyset, d)) \frac{\mathbb{1}_{t,d}}{\sum_{t' \in T} \mathbb{1}_{t',d}} \tag{8}$$

D. Computation of $\mathcal{I}^q(t)$

We consider the entropy reduction definition of the information gain:

$$\mathcal{I}^q(t) = \mathcal{H}_\mathcal{D}^q - \mathcal{H}_\mathcal{D}^{q+t} \tag{9}$$

Where $\mathcal{H}_\mathcal{D}^q$ is the entropy over the document database $\mathcal{D}$ before proposing the keywords and $\mathcal{H}_\mathcal{D}^{q+t}$, the entropy after adding the term t to the query (after the user selected it). After developing Equation 5b, we get:

$$\mathcal{H}_T^q = \sum_{t \in T} p_T^q(t) (\mathcal{H}_\mathcal{D}^q - \mathcal{H}_\mathcal{D}^{q+t}) \tag{10a}$$

[2]It would be puzzling for the user to be proposed the exact same terms as the ones he just pronounced or typed.

$$\mathcal{H}_T^q = \mathcal{H}_\mathcal{D}^q - \sum_{t\in T} p_T^q(t)\mathcal{H}_\mathcal{D}^{q+t} \tag{10b}$$

$\mathcal{H}_\mathcal{D}^q$ is independent of T. Maximising information expectation $E(\mathcal{I}_T)$ is therefore equivalent to minimising the second term:

$$T_n^* = \underset{T\subset\mathcal{T}^{-q}:|T|=n}{\operatorname{argmin}} \sum_{t\in T} p_T^q(t)\mathcal{H}_\mathcal{D}^{q+t} \tag{11}$$

E. Computation of $\mathcal{H}_\mathcal{D}^{q+t}$

$\mathcal{H}_\mathcal{D}^{q+t}$ is the measure of uncertainty in the targeted document if query $q+t$ is expressed. It is formalised as follows:

$$\mathcal{H}_\mathcal{D}^{q+t} = -\sum_{d\in\mathcal{D}} p^{q+t}(d)\ln p^{q+t}(d) \tag{12}$$

Probability $p^{q+t}(d)$ is assessed thanks to the search engine scores:

$$p^{q+t}(d) = \frac{\mu(q+t,d)}{\sum_{d'\in\mathcal{D}}\mu(q+t,d')} \tag{13}$$

F. What if the user answers "none"?

The user may answer "none" to the n keywords presented to her/him. The dialogue system developer might want to take this possibility into consideration and optimise the keyword accordingly. Equation 11 is therefore updated as below:

$$T_n^* = \underset{T\subset\mathcal{T}^{-q}:|T|=n}{\operatorname{argmin}} \left[\sum_{t\in T} p_T^q(t)\mathcal{H}_\mathcal{D}^{q+t} + p_T^q(\emptyset)\mathcal{H}_\mathcal{D}^{q-T}\right] \tag{14}$$

Where $p_T^q(\emptyset)$ is the probability that no term in T matches the user request, given current query q, and $\mathcal{H}_\mathcal{D}^{q-T}$ is the entropy after observing that none of the terms in T is relevant. $p_T^q(\emptyset)$ and respectively $\mathcal{H}_\mathcal{D}^{q-T}$ can be computed thanks to updates of Equations 6 and respepectively 12 and 13:

$$p_T^q(\emptyset) = \frac{\sum_{d\in\mathcal{D}}\mu(q,d)p_T(\emptyset,d)}{\sum_{d\in\mathcal{D}}\mu(q,d)} \tag{15a}$$

$$\mathcal{H}_\mathcal{D}^{q-T} = -\sum_{d\in\mathcal{D}} p^{q-T}(d)\ln p^{q-T}(d) \tag{15b}$$

$$p^{q-T}(d) = \frac{\mu(q-T,d)}{\sum_{d'\in\mathcal{D}}\mu(q-T,d')} \tag{15c}$$

Where $\mu(q-T,d)$ is also an update from Equation 3:

$$\begin{aligned}\mu(q-T,d) &= \mu(q,d) - \mu(T,d) \\ &= \sum_{t\in q}\xi_t w_{t,d}w_{t,q} - \sum_{t'\in T}\xi_{t'}w_{t',d}w_{t',q}\end{aligned} \tag{16}$$

V. Complexity

A. Complexity computation

The search algorithm complexity is $\mathcal{O}(|q||\mathcal{D}|)$. The learning algorithm is performed offline and complexity is not a major issue. On the opposite, the keyword selection is time consuming and is required real time: we have to consider $\mathcal{O}(|\mathcal{T}|^n)$ subsets, which means as many information gain expectation calculation. Fortunately, it is possible to compute in $\mathcal{O}(|\mathcal{D}|)$ for each term t their salience in the document database and to build a subset $\tau\subset\mathcal{T}$ of reasonable cardinality with eligible keywords (see Subsection V-B for more details). Let us have a look to each term in Equation 14.

- The n computations of $p_T^q(t)$ are $\mathcal{O}(n|\mathcal{D}|)$, and are T-dependent.
- The n computations of $\mathcal{H}_\mathcal{D}^{q+t}$ are $\mathcal{O}(|\mathcal{D}|)$, and are T-independent.
- The computation of $p_T^q(\emptyset)$ is $\mathcal{O}(n|\mathcal{D}|)$, and is T-dependent.
- The computation of $\mathcal{H}_\mathcal{D}^{q-T}$ is $\mathcal{O}(|\mathcal{D}|)$, and is T-dependent

The global complexity of the keyword selection algorithm is:

$$\begin{aligned}\mathcal{C} &= \mathcal{O}(n^2|\tau|^n|\mathcal{D}|+n|\mathcal{D}|+n|\tau|^n|\mathcal{D}|+|\tau|^n|\mathcal{D}|) \\ &= \mathcal{O}(n^2|\tau|^n|\mathcal{D}|)\end{aligned} \tag{17}$$

B. Salience-based construction of τ

To evaluate the salience of a term t in a documents set $\mathcal{D}$, constrained by a query q, we propose to measure the information gain expectation for $n=2$ and $T=\{t,\neg t\}$. This computation complexity is low: $\mathcal{O}(|\mathcal{D}|)$ and can play the elimination role: we keep the $\sqrt[n]{|\mathcal{T}|}$ highest information gain expectations. It is still equivalent to minimise the expected resulting entropy as states Equation 11, which we update here:

$$\begin{aligned}\mathcal{H}_{\{t,\neg t\}}^q = &\, p_{\{t,\neg t\}}^q(t)\mathcal{H}_{\{t,\neg t\}}^q(t) \\ &+ (1-p_{\{t,\neg t\}}^q(t))\mathcal{H}_{\{t,\neg t\}}^q(\neg t)\end{aligned} \tag{18}$$

This assessment may still be expensive since we have to make it for every $t\in\mathcal{T}$. For very large databases, it is possible to sample the document base to have an estimation. The $\sqrt[n]{|\mathcal{T}|}$ terms that minimise entropy constitute τ. Therefore, we guarantee that the global complexity of the keyword selection algorithm is:

$$\mathcal{C} = \mathcal{O}(|\mathcal{D}||\mathcal{T}|+n^2|\tau|^n|\mathcal{D}|) \tag{19a}$$

$$\mathcal{C} = \mathcal{O}(n^2|\mathcal{D}||\mathcal{T}|) \tag{19b}$$

There remain several computational optimisation tracks to reduce the complexity such as Latent Semantic Analysis [13] to reduce the cardinality of $\mathcal{T}$ or document sampling to reduce the $|\mathcal{D}|$ factor in Equation 17.

VI. Implementation

The Content Finder AssistanT (CFAsT) implements the model and the algorithms as described in Section II. It is a tool for generating text dialogue applications, which relies on Disserto, the Orange generic spoken and multimodal dialogue technology. Disserto [14] is a software suite that enables the creation of automata-based dialogue application.

A. Architecture

The whole CFAsT architecture is sketched in Figure 3.

1) Dialogue system generation: The low part of the graph describes the generation of the application. The CFAsT configuration is the only mandatory design to generate a first baseline application. It includes the information for the SQL and TiLT connections, the stop words list and stemming dictionary paths, and finally the dialogue messages to play to the user (greeting, search result presentation, content presentation, and validation question). It is possible to specify handcrafted behaviours at this step by adding shortcuts and designing semantic analyser rules. These are templates of improvement 1 in Subsection II-B list. Afterwards, the CFAsT tool generates a Disserto project that can be compiled directly into a .war to be deployed on a Java EE application server. Alternatively, it is possible to use the Disserto suite to modify the application, for instance item 2 in Subsection II-B. It is noticeable that the dialogue application is generated independently from the content base.

2) The indexation architecture: The CFAsT tool is also used to build the SQL database (upper part of Figure 3). It takes the already parsed contents, builds the normaliser and indexes the contents in a PostgreSQL database[3]. The independence between the application and the content is what guarantees its dynamics (addition, substitution and deletion of contents).

The parsing cannot be made automatically: sometimes, we need to parse only designated sections of a document, or a specific text in a web page. The parsing part is thus included in the CFAsT tool. It has to be developed and performed separately. The parsing relies on several free libraries: Apache POI[4] for Microsoft Office documents, Apache PDFBox[5] for PDF documents, java-libpst[6] for Outlook email storage, JSoup[7] for html parsing, and Java MHT Unpack[8] to unpack MHT files.

The following step in indexation is the normalisation. As introduced in Subsection III-A, it relies on different sources, that are mostly language dependent, but can also be optimised accordingly to the application domain.

The spell checker performs a syntactic analysis and corrects any kind of spelling mistake: typo, grammatical error, lexical error, space correction, SMS language, ...In CFAsT, the text input's spelling is corrected by TiLT [10], through http requests. Our first prototypes revealed that some domains used acronyms that were improperly and intermittently corrected by TiLT. It is possible to tune TiLT to correct those unexpected behaviours. CFAsT tools include .

The stemming dictionary regroups word inflexions (such as plural or conjugation) in a common stem. It can also be used as a dictionary for synonyms, simply by designing a common stem for two synonyms. Therefore, its tuning enables improvements 3 and 4 in Subsection II-B list. This dictionary is .csv file with two columns, the first with the inflexion and the second with the stem.

The stop words list gathers utility stems of the language and the domain that should be ignored. The stop words list is loaded from a 1-column .csv file. As the stemming dictionary, it contributes to improvement 3.

The normaliser object is shaded half yellow and orange because, as we have just seen, it is required offline to index the contents, but also online to normalise the use queries according to the same pattern. The normaliser can be updated offline since stemming dictionary and stop words are compiled in an SQL base as well.

3) The artificial intelligence implementations: This subsection deals with the implementation of the algorithms described in Subsections III-B and IV.

The other yellow/orange shaded object is the discriminative keyword selection which embeds the algorithm described in Subsection IV. Indeed, in order to limit computational load on the server, discriminative keywords for a large range of the most common queries are preprocessed offline and stored in an SQL base. The discriminative keywords for new queries, that are computed online, are also saved into the keywords memory SQL table. There is also the possibility to override the keyword selection with predefined keywords. This answers item 5 in the improvement list of Subsection II-B.

The search engine module simply applies Formula 3 using ξ_t that are stored in the search engine parameters table and the tf-idf weights $w_{t,d}$ that are stored in the reverse index table.

And finally, the learning algorithm (SVM) module uses the user clicks to optimise the search engine parameters thanks to Java Native Interface SVM$^{\text{light}}$ library[9] [11].

B. Prototypes

A wide range of prototypes have been automatically generated with the CFAsT tools. Table I takes an inventory of them and presents their main characteristics: number of contents, numbers of terms, average number and standard deviation of different terms in a content. Six of them are made available for a limited lapse of time. Not a lot of effort has been made to design the assistant presentation for the moment. It is worth noticing that it works exclusively on Mozilla Firefox.

The choice of domain, content type and size are completely fortuitous. In fact, the prototypes have been built according to cooperation opportunities. This explains that they are all in French. Luckily, except for the language, they are diverse and enable to emphasise various aspects.

Forum Orange is the biggest content base we had to deal with. The keyword selection algorithm is still fast enough, but

[3]http://www.postgresql.org/

[4]http://poi.apache.org/

[5]http://pdfbox.apache.org/

[6]https://code.google.com/p/java-libpst/

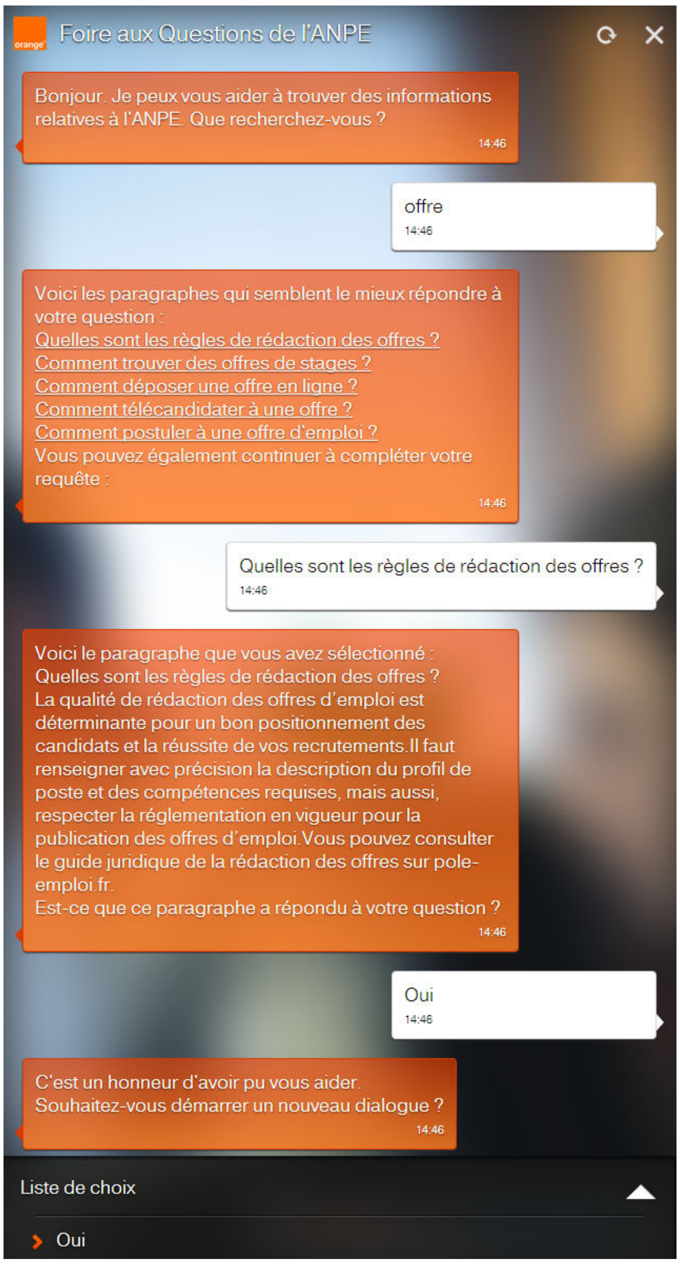
[7]http://jsoup.org/

[8]http://www.chembal.com/java-applications/java-mht-unpack-library

[9]http://svmlight.joachims.org/

Name	Link	Content type	Topic	Size		Contents length	
				Contents	Terms	Average	STD
FAQ ANPE	http://dialogue.orange-labs.fr/ANPEG1R2C2/webClient.html	Q/A	Job Search	160	2368	63	33
Orange Forum	http://dialogue.orange-labs.fr/OrangeEntraideForumG2R1C0/webClient.html	Forum thread	Telecom	6934	31637	86	82
Banque Postale	http://dialogue.orange-labs.fr/LaBanquePostaleG0R0C0/webClient.html	Web page	Retail bank	232	4721	152	101
Orange BS	no	File & mail	Marketing	817	11282	174	225
Lannion	http://dialogue.orange-labs.fr/LannionG0R0C1/webClient.html	Web page	Touristic info	210	5550	107	60
resto.fr (22)	http://dialogue.orange-labs.fr/Resto22G0R0C0/webClient.html	Webpage	Restaurant	253	818	12	7
Traffic laws	no	Article	Traffic laws	210	2675	61	33
AuJardin TV	no	Video	Gardening	294	2692	23	17

TABLE I: Eight french CFAsT text dialogue applications. Five links are provided.


Foire aux Questions de l'ANPE

Bonjour. Je peux vous aider à trouver des informations relatives à l'ANPE. Que recherchez-vous ?
14:46

offre
14:46

Voici les paragraphes qui semblent le mieux répondre à votre question :
Quelles sont les règles de rédaction des offres ?
Comment trouver des offres de stages ?
Comment déposer une offre en ligne ?
Comment télécandidater à une offre ?
Comment postuler à une offre d'emploi ?
Vous pouvez également continuer à compléter votre requête :
14:46

Quelles sont les règles de rédaction des offres ?
14:46

Voici le paragraphe que vous avez sélectionné :
Quelles sont les règles de rédaction des offres ?
La qualité de rédaction des offres d'emploi est déterminante pour un bon positionnement des candidats et la réussite de vos recrutements.Il faut renseigner avec précision la description du profil de poste et des compétences requises, mais aussi, respecter la réglementation en vigueur pour la publication des offres d'emploi.Vous pouvez consulter le guide juridique de la rédaction des offres sur pole-emploi.fr.
Est-ce que ce paragraphe a répondu à votre question ?
14:46

Oui
14:46

C'est un honneur d'avoir pu vous aider.
Souhaitez-vous démarrer un nouveau dialogue ?
14:46

Fig. 2: CFAsT implementation for ANPE.

it comes to border line as it can take several seconds on some queries. It is a content base that is not well controlled. For instance, some threads might have the exact same title or some can include abuse. Also, it is growing on a daily basis.

FAQ ANPE is a Frequently Asked Question list that have been integrated to the assistant. It shows that, even on a small content base, the algorithm can find a relevant discriminant keyword selection, by (awkwardly) asking whether the user is an employer or looking for a job. On this CFAsT application, we also illustrated formatted ways to add dialogue strategies. For instance, asking for “mto” will result in a dedicated joke and a refocusing on the assistant domain. There is also a disambiguation strategy that was implemented if the user asks: “créer un espace”.

The Orange Business Service application is the one that has the most focused our attention up to now, but we cannot make it available for obvious confidentiality reasons. Its particularity is to index very heterogeneous contents: web pages, mails and various kinds of files (Excel, Word, Powerpoint and PDF files). The biggest issue is to deal with contents of variable size: a one-liner mail and a document of one hundred pages.

But the dialogue application that defines best the limits of the fully text-based approach is the restaurant finder (resto.fr). The site is presented with a short form including the town, the price range, the cooking category and some additional information. CFAsT algorithm cannot compute distance between instances. For example, a 12€ menu is acceptable when the user specifies a 15€ budget. Also it does not have a hyperonymy structure to understand that Japanese food is okay when the user asks for an Asian restaurant. The form-based assistants must be handled specifically. Form-based dialogues, that are considered as the easiest ones for MDP-based and handcrafted systems are among the hardest for this search-based model. This demonstrates their complementarity.

VII. Discussion

In introduction, we claimed that most reservations expressed in [19], [20] were alleviated by our approach on automatic generation of dialogue systems. The main one concerns the *VUI completeness* required in industry, and that “entails that all possible combinations of user inputs and conditions need to be anticipated”. Since our model is looping on a single research dialogue state, it does guarantee that every situation is anticipated. Moreover, users are used to search engines and know they work by searching for documents containing a maximum of query terms. Another reservation relates to the need of a large dialogue corpus, which is often missing when building a first version of a dialogue system. Our model does not require any: only the contents. And finally, a third reservation[10] relies on the loss of control with the reinforcement learning approach. Our model can be overloaded with handcrafted rules, granting the developers a full control on the so-built application.

But, is it really dialogue? Isn't it an advanced way to perform search? Obviously, the task is information retrieval. Although, it is performed through dialogue. The dialogue helps the user to formulate his query and to navigate through the hits. Additionally, the dialogue helps to write a user experience story and therefore to learn how to better interact in the future.

Okay, but the dialogue remains very basic. All the dialogue systems presented in this paper were entirely generated. They are beta versions that can be deployed to gather first user experience. They look like fancy search engines. But, it is advised to add human or semi-automated design. At first, it becomes a search engine with some hard coded dialogue strategies. Then, it includes more and more dialogue strategies, until being similar to a dialogue system that uses a search engine as a parachute: when the dialogue engine fails to find an answer, the search engine can find an answer. The whole search-engine/dialogue-system spectrum is possible.

Towards more and more automation. Unlike Reinforcement Learning dialogue systems, CFAsT tools do not pretend to generate systems that outperform handcrafted dialogue systems. This article only aims to demonstrate that it is possible to generate a basic dialogue application with no specific work. It is a first stone in this direction. Our future work on this topic includes to enable multi-field queries with a topology that is optimised over experience, thanks to a recommender system [17] and to automatically propose dialogue strategies to a human designer, whose role is to formulate the questions and answers.

References

[1] A. Abella, J. Wright, and A. Gorin, "Dialog trajectory analysis," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 1, May 2004, pp. 441–444.

[2] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer networks and ISDN systems*, vol. 30, no. 1, pp. 107–117, 1998.

[3] S. Chandramohan, M. Geist, F. Lefevre, and O. Pietquin, "User simulation in dialogue systems using inverse reinforcement learning," in *Proceedings of Interspeech*, 2011, pp. 1025–1028.

[4] C. Cortes and V. Vapnik, "Support vector machine," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[5] I. Dhillon, S. Mallela, and D. Modha, "Information-theoretic co-clustering," in *Proceedings of ACM SIGKDD*. ACM, 2003, pp. 89–98.

[6] L. El Asri, R. Laroche, and O. Pietquin, "Reward function learning for dialogue management." in *Proceedings of STAIRS*, 2012, pp. 95–106.

[7] D. Ferrucci, "Introduction to "this is watson"," *IBM Journal of Research and Development*, vol. 56, no. 3.4, pp. 1:1–1:15, May 2012.

[8] W. Gross, J. Levine, and G. Kong, "Search engine with suggestion tool and method of using same," 2006, uS Patent App. 11/404,944.

[9] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[10] J. Heinecke, G. Smits, C. Chardenon, E. De Neef, E. Maillebuau, and M. Boualem, "Tilt: plate-forme pour le traitement automatique des langues naturelles," *Traitement automatique des langues*, vol. 49, no. 2, pp. 17–41, 2008.

[11] T. Joachims, "Optimizing search engines using clickthrough data," in *Proceedings of ACM SIGKDD*, 2002, pp. 133–142.

[12] Q. Jones, G. Ravid, and S. Rafaeli, "Information overload and the message dynamics of online interaction spaces: A theoretical model and empirical exploration," *Information systems research*, vol. 15, no. 2, pp. 194–210, 2004.

[13] T. Landauer, P. Foltz, and D. Laham, "An introduction to latent semantic analysis," *Discourse processes*, vol. 25, no. 2-3, pp. 259–284, 1998.

[14] R. Laroche, P. Bretier, and G. Putois, "Enhanced monitoring tools and online dialogue optimisation merged into a new spoken dialogue system design experience," in *Proceedings of Interspeech*, Chiba (Japan), September 2010.

[15] E. Levin, R. Pieraccini, and W. Eckert, "Using markov decision process for learning dialogue strategies," in *Proceedings of ICASSP*, 1998. [Online]. Available: citeseer.ist.psu.edu/article/levin98using.html

[16] C. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge university press Cambridge, 2008, vol. 1.

[17] F. Meyer and F. Fessant, "Reperio: A generic and flexible industrial recommender system," in *Proceedings of IEEE/WIC/ACM WI-IAT*. IEEE Computer Society, 2011, pp. 502–505.

[18] R. Ortega, J. Avery, and R. Frederick, "Search query autocompletion," 2003, uS Patent 6,564,213.

[19] T. Paek, "Reinforcement learning for spoken dialogue systems: Comparing strengths and weaknesses for practical deployment," 2006. [Online]. Available: citeseer.ist.psu.edu/singh99reinforcement.html

[20] T. Paek and R. Pieraccini, "Automating spoken dialogue management design using machine learning: An industry perspective," *Speech Communication*, vol. 50, pp. 716–729, 2008.

[21] R. Pieraccini and D. Suendermann, "Data-driven methods in industrial spoken dialog systems," in *Data-Driven Methods for Adaptive Spoken Dialogue Systems*. Springer, 2012, pp. 151–171.

[22] O. Pietquin, M. Geist, and S. Chandramohan, "Sample efficient on-line learning of optimal dialogue policies with kalman temporal differences," in *Proceedings of IJCAI*. AAAI Press, 2011, pp. 1878–1883. [Online]. Available: http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-314

[23] G. Putois, R. Laroche, and P. Bretier, "Online reinforcement learning for spoken dialogue systems: The story of a commercial deployment success," in *Proceedings of SIGDIAL*, Tokyo (Japan), September 2010.

[24] G. Salton, A. Wong, and C. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[25] J. Shanahan and G. Grefenstette, "Document-centric system with auto-completion," 2004, uS Patent 6,820,075.

[26] C. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[27] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, March 1998. [Online]. Available: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&amp;path=ASIN/0262193981

[28] J. Williams, "The best of both worlds: Unifying conventional dialog systems and POMDPs," in *Proceedings of the International Conference on Speech and Language Processing (ICSLP)*, 2008.

[29] J. Williams and S. Young, "Scaling up POMDPs for dialog management: The summary POMDP method," in *Proceedings of IEEE ASRU*, 2005, pp. 177–182.

[30] J. Wright, D. Kapilow, and A. Abella, "Interactive visualization of human-machine dialogs," in *Proceedings of the European Conference on Speech Communication and Technologies (Interspeech'07)*, Lisbon (Portugal), September 2005, pp. 2517–2520.

[31] S. J. Young, "Probabilistic methods in spoken-dialogue systems," in *Philosophical Transactions of the Royal Society*, ser. Royal Society of London Philosophical Transactions Series A, vol. 358, Apr. 2000, pp. 1389–1402.

[10]Other reservations are about the reinforcement learning model and the optimisation claim, both of which are out of scope in this paper.

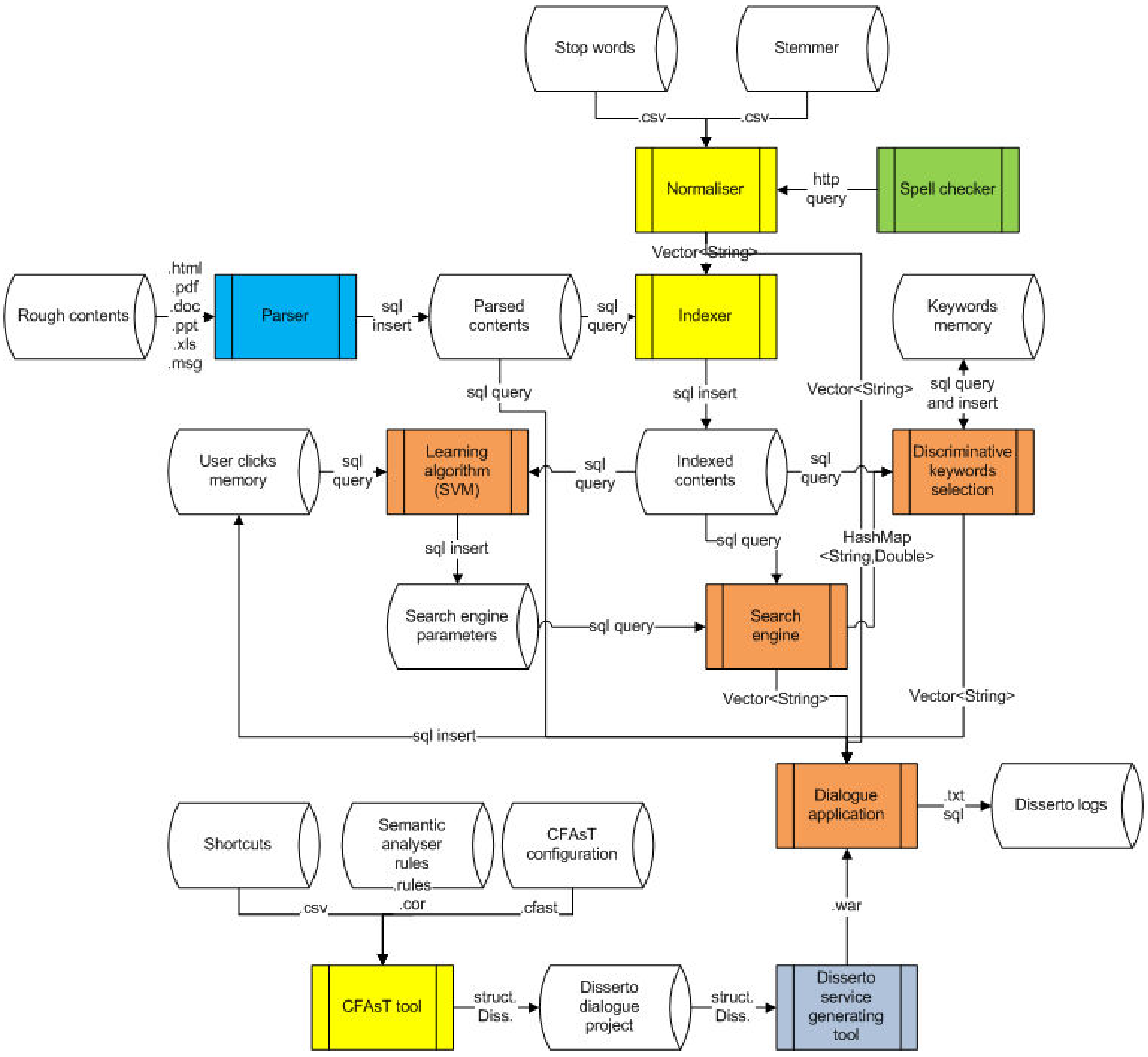


Fig. 3: CFAsT architecture: orange boxes are CFAsT runtime (online) modules, yellow boxes are CFAsT offline tools, the green box is TiLT module, and the teal box is Disserto. The blank boxes are data. The ones that are used online are stored in a PostgreSQL database.