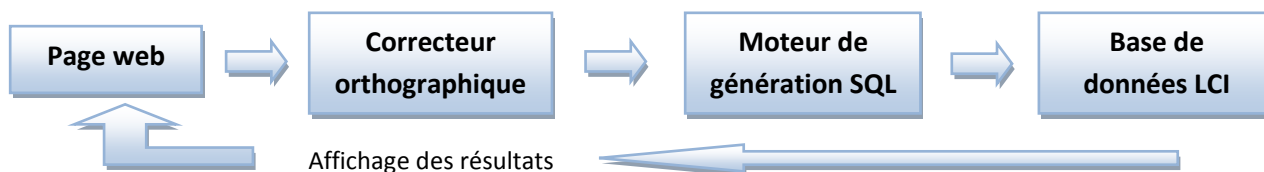


## Caractéristiques principales de l'application

### Architecture

L'application est répartie en plusieurs modules qui ont été intégrés un à un à la fin, après tests unitaires :

- ❖ La servlet Java qui permet l'interface entre l'utilisateur et l'application (mots-clés de la recherche et affichage du résultat)
- ❖ Le correcteur orthographique qui s'occupe de corriger les fautes de frappe et d'orthographe de la requête entrée par l'utilisateur
- ❖ Le moteur de génération du SQL qui transforme la requête émise par l'utilisateur en langage naturel en langage universel de requête (SQL) afin d'interroger la base de données
- ❖ La base de données qui contient les informations des pages et articles LCI indexés en début de projet



### Principes de fonctionnement

L'utilisateur entre donc une phrase formulée en langage naturel afin de rechercher un ou des articles contenus dans le corpus LCI dans un formulaire WEB. La servlet Java récupère cette chaîne de caractères, la transmet au correcteur orthographique qui génère en sortie une chaîne corrigée, c'est-à-dire dépourvue de ses fautes d'orthographe et de frappe d'origine. Ensuite cette chaîne corrigée est transmise au module de génération SQL qui procède en 2 phases : la première génère à partir de cette chaîne un arbre syntaxique (AST) représentant les concepts de la requête de l'utilisateur ; dans un second temps, cet arbre est transformé en arbre SQL équivalent à la requête. Cet arbre est ensuite post-traité pour gérer certains cas délicats à traiter dans le module de génération SQL à proprement parler et en sort une chaîne SQL correspondant à la requête entrée par l'utilisateur au départ. Cette requête SQL est enfin exécutée sur le serveur PostgreSQL de l'UV via une API Java : les résultats sont ensuite affichés via la servlet Java dans un frame de la page WEB dans laquelle l'utilisateur a entré sa requête initiale.

## Performances

### Types de phrases reconnues

- ❖ « Je veux les articles qui parlent de bush et des usa »
- ❖ « Donne moi les pages écrites par dsstrauss@tf1 et qui parlent du pape »
- ❖ « Je voudrais tous les articles sur le thème de la guerre et contenant CIA »
- ❖ « Quelle est la liste des articles parus en février 2005 »

### Traitement des termes de la requête

Nous avons adapté la grammaire à l'application et laissé le module de morphologie et le lexique génériques.

### Paramètres statiques

SELECT	OBJET	CARACTERE
vouloir ; quel est ; retourne ; donne moi ; combien	article ; page ; ensemble des articles ; liste des pages	sur le thème ; écrits par ; parus en ; contenir ; qui parlent ; traitant

### Paramètres dynamiques

**Variable** : un mot du lexique contenu dans un article ou une page / un nom propre / une adresse mail d'un auteur (la variable est interprétée différemment selon le contexte dans lequel elle est utilisée)

**Date** : sous format jour mois année / mois année / année avec le nom des mois en toutes lettres

Nous avons fait le choix arbitraire de ne pas proposer de correction à l'utilisateur à la manière de Google par exemple et de ne conserver donc que la correction la plus plausible dans les algorithmes (Levenshtein).

Enfin, les résultats sont présentés dans un tableau HTML avec des liens cliquables s'il s'agit de pages LCI retournées.

## Critiques de notre application

### Problèmes rencontrés

- Ajuster les paramètres des algorithmes tels que dans le correcteur orthographique n'est pas évident car il n'y a pas vraiment de bonne valeur
- Configurer à chaque fois Tomcat afin qu'il détecte la servlet (fichier web.xml)
- Prendre garde aux permissions d'accès aux fichiers de l'application (groupe Apache)
- Problème de passage des caractères accentués entre la page HTML et le correcteur orthographique, ce qui nous produit une requête corrigée erronée et par conséquent ne renvoie aucun résultat.

### Limites de performance et de robustesse

Dans ce projet, il a fallu combiner au sein d'une même application, des algorithmes de correction orthographique, d'analyse de langage et de traitement de base de données. Il a fallu équilibrer toutes ces composantes afin d'arriver à un résultat satisfaisant pour l'utilisateur. Ceci peut poser problème dans le cas où la correction orthographique est trop importante car la requête finale va porter sur des éléments différents de ce que l'utilisateur désirait initialement (même si l'on voulait juste corriger quelques erreurs de frappe).

### Pistes d'améliorations de l'application

Nous pensons qu'il a fallu plus de temps pour concevoir la partie analyse syntaxique afin de pouvoir répondre à plus de requêtes du corpus de questions. Ici, nous nous sommes retrouvés avec un système d'analyse qui gère quelques types de requêtes que nous avons jugé importantes.

En outre, notre système ne supporte pas les requêtes comportant des critères de tri. Ceci est dû à la non prise en compte de ces requêtes lors de notre conception. Nous avons préféré simplifier la liste des questions au départ et intégrer des questions de plus en plus complexes au fur et à mesure.

Voici par type de questions les améliorations qu'il faudrait envisager pour améliorer le système :

#### Le comptage d'articles :

Une amélioration de la grammaire aurait permis de traiter les requêtes de type :

- *Combien d'articles évoquent Berlusconi et le fascisme ?*
- *Combien d'auteurs différents ont écrit des articles le 24 octobre 2005 ?*
- *Quels sont les auteurs qui ont écrit le plus d'article ce mois-ci ?*

Ces requêtes pourraient être traitées avec une petite adaptation de notre analyse d'arbre. Dans le corpus de questions beaucoup de requêtes sont basées sur le comptage d'articles, d'auteurs...

Notre grammaire permet aujourd'hui de récupérer des comptages (« count » SQL) uniquement pour les requêtes simples, sans termes « ET » ni « OU ». Par exemple : « combien d'articles parlent de Bush » fonctionne mais pas « combien d'articles parlent de Bush et du pape ». Ceci est dû à notre phase de post-traitement de l'arbre SQL renvoyé par notre grammaire et pourrait être amélioré par la suite.

#### Dates complexes:

Nous appelons comme ceci les requêtes de type :

- *Quels sont les auteurs qui ont écrit le plus d'article ce mois-ci ?*
- *Donne-moi la liste des articles où l'on a parlé du président des états Unis depuis 3 semaines.*

La prise en compte de ce type de requêtes n'est pas forcément compliquée (peut-être un peu longue), mais elle pourrait donner une réponse à de nombreuses requêtes supplémentaires du corpus de questions. Actuellement nous ne prenons en compte que les dates jour mois année / mois année / année.

#### Requêtes complexes :

Ce sont les requêtes de type :

- *Quand Bush s'est-il opposé à l'euthanasie ?*
- *De quoi parlait-on le plus du 27 mars au 4 avril 2005 ?*
- *Comment puis-je contacter l'auteur Hervé Moustache ?*
- *Parle-t-on dans le même jour d'un attentat en Irak et du pape dans une même Une ?*

Ces requêtes seraient très coûteuses car elles nécessiteraient une analyse sémantique. Et cela sort du cadre de ce projet.