Fabien BROSSIER
Christophe DESCLAUX
SI5 Spécialité KIS

POLYTECH NICE-SOPHIA

Web Sémantique

Rapport de Projet

Document : TabooGeek - Rapport de Projet

Matière : Web Sémantique

Enseignants : M. Olivier CORBY
M. Fabien GANDON
Mme. Catherine FARON-ZUCKER

Créateurs : M. Fabien BROSSIER
M. Christophe DESCLAUX

Mardi 21 février 2012

I. Présentation du projet

Ce jeu est une version informatique du jeu « Taboo » dont les règles ont été quelque peu modifiées. Ces modifications sont dues aux contraintes techniques et temporelles.

1. But du jeu original

Le jeu original se joue au moins à 4, soit 2 équipes de 2 personnes. On peut au delà de cette limite augmenter le nombre d'équipes et/ou le nombre de personnes par équipe à volonté. Le but du jeu est le suivant :

- Un joueur de la première équipe doit faire deviner un mot à ses coéquipiers en moins d'une minute.
- Pour cela il tire une carte sur lequel est marqué le mot à faire deviner, ainsi que 5 autres mots, les mots « interdits ».
- Le joueur doit ensuite faire prononcer le mot à deviner à n'importe lequel des membres de son équipe sans prononcer 1 seul des 5 mots interdits.
- Il n'a le droit que de parler, les mimes et autres gestes sont interdits.
- Si un joueur de l'équipe a deviné le mot correct dans le temps imparti, l'équipe gagne 1 point. Les équipes se succèdent et au bout d'un certain nombre de tours, l 'équipe ayant le plus de points gagne le jeu.

Note : Tout au long de ce rapport, nous désignerons les joueurs par « Joueur 1 » et « Joueur 2 ». Il est entendu ici que le joueur 1 doit faire deviner le mot et le joueur 2 doit le deviner.

2. But du jeu adapté

Nous avons quelque peu modifié les règles du jeu pour pouvoir l'adapter numériquement dans le temps imparti. Il aurait été pertinent dans un projet comme celui-ci de fournir la possibilité de jouer en réseau.

Le premier joueur pourrait voir s'afficher le mot à deviner ainsi que la liste des mots interdits et taperait le texte permettant à un joueur distant de deviner le mot en question. Cette idée a été abandonnée pour 2 raisons :

- Dans la réalité, alors que le premier joueur dicte les indices permettant au second joueur de découvrir le mot, ce dernier entend les paroles en temps réel. La notion de jeu en réseau implique que cette information est transmise de façon asynchrone et par conséquent, il est difficile de calculer le temps exact utilisé pour deviner le mot.
- La création d'un jeu en réseau nécessite des technologies supplémentaires à mettre en place telle que JMS, et implique par conséquent un temps de développement relativement long, non envisageable dans la durée actuelle du mini-projet.

Nous avons modifié quelques règles du jeu, par souci de simplicité technique, ou afin de mettre en application certains aspects directement liés au Web Sémantique. Les modifications sont les suivantes :

- Le joueur peut choisir le mot qu'il souhaite faire deviner à son adversaire. Ce mot est entré dans un champ de texte, et une fenêtre proposant les mots disponibles dans la base s'affiche si le joueur appuie sur la flèche du bas.
- La durée maximum pendant laquelle le deuxième joueur peut deviner le mot a été augmentée à 1 minute et 30 secondes.
- Aucun mécanisme ne compte le dépassement temporel attribué à l'équipe. Les points ne sont pas comptabilisés, il appartient aux joueurs de tenir leurs comptes.
- Lorsque le deuxième joueur découvre le mot qu'il fallait deviner, le premier joueur peut inscrire quel a été l'indice donné qui a permis la découverte. Cet indice sera ensuite stocké dans la base pour améliorer le jeu.

Le déroulement du jeu est donc le suivant :

- Le joueur 1 inscrit le mot à faire deviner dans un champs de texte.
- La page lui renvoie la liste des mots interdits ainsi qu'un compte à rebours qui lui permet de vérifier qu'il lui reste toujours du temps pour faire deviner le mot à son équipe.
- Le joueur 2 essaye de deviner le mot et le joueur 1 adapte son discours pour se faire.
- Lorsque le joueur 2 a deviné le bon mot, le joueur 1 inscrit l'indice qui a permis au joueur 2 de découvrir le bon mot dans un champ de texte ; le système l'ajoutera ensuite à la base de données.

2. Technologies utilisées

Nous avons utilisé 3 technologies pour ce projet : 4store, Ruby on Rails, et HTML5. 4Store nous a permis de gérer les données du projet, et Ruby on Rails fait le lien entre les couches « données » et « vue », générée en HTML5.

1. Ruby on Rails

Ruby on Rails est un serveur créé en Ruby permettant de créer des applications WEB respectant le pattern MVC. Dans ce projet, nous n'utilisons pas la partie « Modèle » fournie par Ruby on Rails puisque les bases de données fournies sont en SQL.

Dans le controlleur nous effectuons donc une connexion vers un serveur distant 4store hébergeant une base de données RDF. Nous transmettons des requêtes SPARQL et récupérons un résultat RDF que nous analysons et affichons à l'écran.

2. 4store

4store est un serveur de triplets RDF fournissant un accès limité aux données. Si la lecture et la création de données s'effectue assez simplement, la suppression et la mise à jour sont difficiles voire impossibles puisque la norme SPARQL n'a pas été implémentée dans sa totalité.

4store permet de créer un endPoint en ligne rapidement, ce qui nous a permis de concentrer notre travail sur les requêtes à effectuer plutôt que sur la configuration du serveur. De plus, cela permet à d'autres développeurs d'utiliser également notre serveur pour leurs requêtes.

Nous avions au départ choisi cette technologie pour s'affranchir des contraintes de librairies liées au langage puisque 4store fournie des librairies d'accès dans tous les langages, notamment Ruby.

3. HTML5

Outre la structure et l'aspect général du site, nous utilisons HTML5 pour afficher la liste des mots disponibles à faire deviner. Cette fonctionnalité est actuellement disponible sur les dernières versions de Firefox uniquement, et c'est pourquoi nous conseillons l'utilisation de notre site avec ce navigateur.

En considérant le temps accordé au projet et les objectifs principaux à remplir, nous avons pensé que cette fonctionnalité était minimale et c'est pourquoi nous n'utilisons pas un système plus complexe compatible multi-navigateurs.

Lorsque l'utilisateur tape une lettre ou appuie sur la flèche du bas, la liste s'affiche et propose donc d'auto-compléter le champ selon le mot souhaité par l'utilisateur.

3. Données SKOS et requêtes SPARQL

1. Les requêtes SPARQL

La liaison avec nos données s'effectue par l'intermédiaire de requêtes SPARQL. 4Store gère mal, à priori, les préfixes. Par conséquent, si la requête effective n'utilise pas les préfixes, nous les avons tout de même remplacés ici pour davantage de visibilité. Vous pouvez visualiser ci-dessous un exemple de requête utilisée :

```
SELECT ?prefLabel ?concept ?altLabel ?element ?weight WHERE {
      ?concept skos:prefLabel ?prefLabel
      ?concept tabooGeek:element ?element
      ?element skos:altLabel ?altLabel
      ?element tabooGeek:weight ?weight
      FILTER regex(?prefLabel, @word.capitalize)
} ORDER BY DESC(?weight) LIMIT 5
```

La requête ci-dessus permet de récupérer les différents mots interdits et leur valeur de pondération associé au mot tapé par l'utilisateur. La pondération du mot est un élément déterminant dans l'évolution du jeu et sera davantage détaillée dans la suite du rapport.

2. Le stockage des données

Le stockage des données sur le serveur 4store s'effectue dans un fichier RDF. Les mots à deviner étant des concepts, nous utilisons beaucoup la norme SKOS. Les informations stockées sont les suivantes :

- La définition d'un concept, stockée ici uniquement à titre informatif et dans un souci d'évolutivité. Ce stockage peut-être intéressant pour d'autres utilisateurs ou développeurs accédant à notre base.
- Au moins un concept parent, qui doit obligatoirement être le concept `http://tabooGeek.zouig.org/#Informatique`
  Cela garantit que les nouveaux concepts stockés sont liés au domaine informatique.
- Le label préféré, qui constitue le mot décrivant l'entité. C'est ce label que l'on récupère pour définir la liste des mots que l'on peut deviner.
- Des ressources de type « élément », que nous avons défini nous-même, et qui contiennent à la fois un label alternatif, et une valeur de pondération.

Notre base de données contenait à la base une poignée de mots pour constituer le jeu de tests. Nous avons par la suite effectuer des requêtes SPARQL complexes sur la base de données DBPEDIA pour récupérer davantage de concepts ayant un rapport avec l'informatique, mais nous obtenons trop d'informations complexes.

En effet, ce jeu étant destiné à un public semi-averti en informatique, nous recherchons des mots simples tels que « informatique », « octet » ou « programmation », et non « Informatique par langage naturel » ou « Internationalisation et localisation » qui est un exemple de ce que renvoie DBPEDIA.

2 exemples de tels fichiers sont visualisables dans le répertoire : /app/assets/skos/

Un exemple de concept est présenté ci-dessous :

```
<skos:Concept rdf:about="http://tabooGeek.zouig.org/#Informatique">

     <skos:definition>L'informatique est le domaine …</skos:definition>
     <skos:prefLabel>Informatique</skos:prefLabel>

     <tabooGeek:element rdf:parseType='Resource'>
          <skos:altLabel>Ordinateur</skos:altLabel>
          <tabooGeek:weight>1</tabooGeek:weight>
     </tabooGeek:element>

          ...

     <tabooGeek:element rdf:parseType='Resource'>
          <skos:altLabel>Donnée</skos:altLabel>
          <tabooGeek:weight>1</tabooGeek:weight>
     </tabooGeek:element>

</skos:Concept>
```

3. La partie serious

La pondération ajoutée aux concepts est un élément déterminant pour notre jeu. En effet, lorsque le joueur 2 a deviné le bon mot, le joueur 1 entre l'indice qui a permis la découverte.du mot dans le champ approprié. Ce mot est ensuite ajouté au concept associé en tant que nouvel élément et sa pondération est placée à 1.

```
     <tabooGeek:element rdf:parseType='Resource'>
          <skos:altLabel>Nouveau mot</skos:altLabel>
          <tabooGeek:weight>1</tabooGeek:weight>
     </tabooGeek:element>
```

Si en revanche le mot existait déjà dans la base, alors sa pondération est incrémentée.

```
     <tabooGeek:element rdf:parseType='Resource'>
          <skos:altLabel>Nouveau mot</skos:altLabel>
          <tabooGeek:weight>1</tabooGeek:weight>
     </tabooGeek:element>
```

Les 5 mots interdits associés à un mot à deviner donné sont générés à partir de la liste des éléments. Une requête SPARQL (celle montrée en exemple ci-dessus) renvoie les 5 mots ayant la pondération la plus élevée. Cela permet donc de compléter au fur et à mesure des parties la base de données du jeu tout en gardant la difficulté du jeu à un niveau correct.

4. Evolutions pour l'avenir

Notre jeu peut être amélioré. La tâche principale serait d'augmenter la taille de la base de données à partir d'un sous ensemble de DBPEDIA. Les fichiers SKOS présents dans le répertoire /app/assets/skos constituent une bonne base de départ.

Néanmoins, il faut effectuer un pré-traitement dessus afin de rajouter les pondérations pour chaque mot. De plus, ces mots là n'ont pas ni définition ni de label alternatif. Nous pensions qu'avec une base plus importante, nous pouvions utiliser les relations de type *skos:related* pour déterminer les mots interdits, mais à l'heure actuelle notre base n'est pas assez peuplée.

La deuxième amélioration à apporter au projet réside dans la possibilité de jouer à plusieurs joueurs. Actuellement un seul joueur utilise l'ordinateur et fait deviner le mot à l'autre joueur. Il serait intéressant de pouvoir permettre un jeu en réseau intranet.

Les joueurs seraient face à face et auraient 2 pages différentes sur leur écran respectif. Le joueur 1 verrait la liste des mots interdits et le joueur 2 aurait un champ de réponse. Le site pourrait signaler de lui-même au joueur 2 si la proposition est correcte ou pas, et pourrait bloquer les pages des joueurs si le temps de réponse est dépassé.

Ce projet nous a beaucoup intéressé dans la mesure où il constitue une mise en pratique des connaissances apprises en Web Sémantique. Néanmoins, si les technologies du web sémantique sont très puissantes, elles nécessitent un jeu de données relativement imposant pour être pleinement exploitables, et par conséquent des ressources temporelles importantes pour sa mise en place, ce qui n'était pas le cas pour ce mini-projet.