TER Master 1 (FMIN200)
Développement d'un Oracle Lexical
Rapport Final

VEYSSIER Julien, BISQUERT Pierre,
PAIVA LIMA DA SILVA Bruno, BELMONTE Rémy
Encadrant : M. Lafourcade

2 juin 2009

Table des matières

Table des figures

1 Introduction

1.1 Présentation du sujet

Dans le cadre du TER de Master première année, nous avons été amenés à développer une série d'applications, toutes axées autour d'un but commun : obtenir des données lexicales de qualité et les utiliser dans une interaction entre un humain et un programme.

Le premier objectif a été de développer un oracle lexical sous la forme d'un outil : LexiTool. L'oracle est une application web qui permet par exemple de retrouver un mot que l'on aurait oublié en se servant d'indices fournis par l'utilisateur. Cet oracle se sert de la base de données lexicales afin de fournir une réponse correcte à l'utilisateur.

Le deuxième objectif a été de développer un jeu de devinette. Ce jeu comporte deux modes :

- Un premier mode où l'utilisateur devra deviner un mot choisi par l'ordinateur. Des mots ou indices lui seront proposés les uns après les autres jusqu'à ce qu'il trouve la bonne réponse. Toute réponse pourra aussi aider à créer des données lexicales.
- Un second mode de jeu où ce sera à la machine de deviner un mot proposé par l'utilisateur. Ce mode de jeu est très proche par son fonctionnement et son principe de l'outil LexiTool.

Le troisième objectif a été d'améliorer l'oracle en permettant à l'utilisateur de proposer un mot en cas d'échec de la part du programme.

Le quatrième objectif était de pouvoir créer ou obtenir des données lexicales en effectuant de la fouille de texte.

L'oracle et le jeu de devinette sont accessibles sur le web dans le but de faire participer un maximum de personnes à l'alimentation de nos données et d'obtenir une grande diversité dans les informations récoltées.

1.2 Inspirations et origines

Les principes qui sous-tendent l'oracle lexical ainsi que son fonctionnement sont multiples et de natures diverses. Nous pouvons retenir trois principales sources d'inspiration :

1.2.1 JeuxDeMots et Pticlic

La première est bien sûr l'outil JeuxDeMots, développé au LIRMM par Mathieu Lafourcade, et dont nous avons récupéré les données lexicales sous la forme d'une sauvegarde de la base de données.

Fig. 1 – Capture d'écran du site web JeuxDeMots

La structure et l'esprit dans lequel JeuxDeMots a été développé nous ont guidé tout au long du projet.

En effet, la structure de la base de données de JeuxDeMots a fortement influencé notre modélisation du problème, puisqu'elle a été notre source de données initiales pour lancer l'oracle.

De plus, outre les aspects purement techniques liés à l'implémentation de la base de données et à la représentation des relations entre les mots par des vecteurs sémantiques, l'outil Pticlic, lui aussi développé au LIRMM par M. Lafourcade, nous a permis d'obtenir des pistes pour rendre le jeu attrayant pour les utilisateurs (nous verrons par la suite que ce point a soulevé de nombreuses questions).

Enfin, le dernier point sur lequel Pticlic et JeuxDeMots nous ont inspiré est la sécurité, afin de mettre au point des mécanismes défensifs contre les utilisateurs malveillants, ou tout simplement inexpérimentés (acquisition de données incorrectes ou non pertinentes essentiellement).

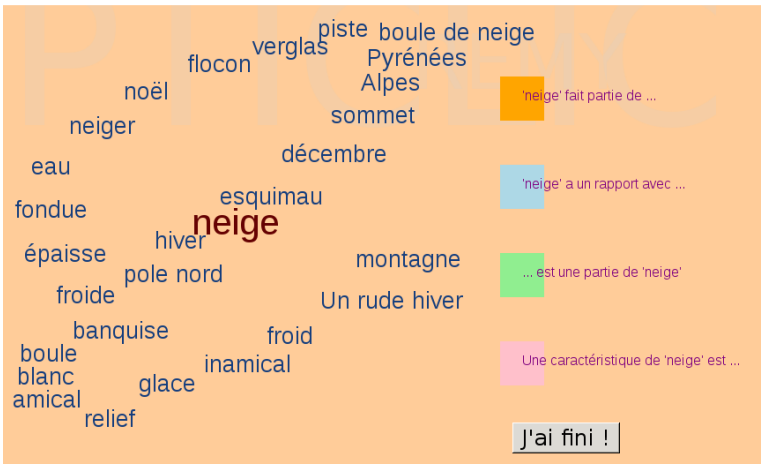

FIG. 2 – Capture d'écran du site web Pticlic

1.2.2 Akinator

Nous avons ensuite été inspirés par le site web Akinator[1].

Akinator est un site web où l'on peut tenter de faire deviner l'identité d'une personne plus ou moins célèbre à laquelle on pense en répondant à une série de questions.

On remarque facilement le lien étroit entre Akinator et l'oracle. En effet, ce dernier est une version étendue d'Akinator dans la mesure où il devra pouvoir deviner virtuellement n'importe quel mot (dans la limite de ses données lexicales).

Akinator a donc été pour nous une grande source d'inspiration, du moins au début, sur certains points, comme notamment les moyens de rendre le jeu attrayant pour les joueurs.

Par la suite, nous avons progressivement abandonné le parallèle avec Akinator pour nous tourner vers un autre jeu, bien plus connu, et qui correspondait à ce que nous cherchions à faire.

[1] http ://www.akinator.com/aki_fr/

FIG. 3 – Capture d'écran du site web Akinator

1.2.3 Pyramide

Notre troisième et dernière grande source d'inspiration aura été le jeu télévisé Pyramide.

En effet, durant ce jeu, les candidats ont pour but de se faire deviner mutuellement des mots choisis au préalable en se donnant des indices (soumis à des contraintes), et en un minimum de temps.

Encore une fois, la ressemblance du concept avec celui de l'oracle est évidente, c'est pourquoi Pyramide a été une grande source d'inspiration pour nous, et ce à de nombreux titres.

Par exemple, il nous a permis d'imaginer les différents modes de jeu ainsi que les méthodes d'attribution des points, ou encore de faire certains choix concernant le fonctionnement de l'oracle (par exemple la façon de deviner un mot).

Il est bien sûr évident que le niveau des joueurs de Pyramide est bien supérieur à celui de l'oracle, cependant plusieurs pistes ont été mises en évidence, comme par exemple l'utilisation, de façon relativement simpliste, des différents types de relations.
Ce problème est par ailleurs comblé par une capacité d'apprentissage illimité et une restitution sans faille.

En ce sens, Pyramide nous a permis d'envisager de nombreuses voies concernant les possibilités de développement de l'oracle, mais aussi de nous fixer des limites quant à ce que nous pouvions et ne pouvions pas faire.

1.3 Les contraintes

Les contraintes ont été, elles aussi, de natures variées, et nous ont orienté dans les choix que nous avons eu à faire de façon déterminante.

La première contrainte, fondamentale, à laquelle nous avons tout d'abord été confrontés a été l'accessibilité de l'oracle : il est en effet important de permettre un accès simple et rapide aux utilisateurs potentiels afin qu'ils soient le plus nombreux possible.

Le choix de le développer comme une application web s'est donc naturellement imposé à nous comme étant le plus pertinent. Ceci nous a de plus permis de tenir à jour notre base de données lexicales plus aisément.

Ce choix d'une application web implique cependant de devoir faire preuve d'une grande vigilance et de prendre en compte la sécurité du site, pour éviter par exemple qu'un utilisateur peu scrupuleux puisse tricher pendant une partie (modification des points ou encore accès illicite à la base de données lexicales).

Ceci aurait en effet risqué de corrompre les données lexicales recueillies mais aussi de limiter le plaisir potentiel des utilisateurs et ainsi réduire la fréquence de leurs passages, et donc la quantité de données recueillies.

En effet, notre travail étant basée sur une étude statistique des données, leur quantité est tout aussi fondamentale pour assurer leur pertinence que leur qualité (peu de données risqueraient de donner des résultats incomplets voire faux).

Les résultats seront enfin transmis à notre encadrant soit sous la forme d'une sauvegarde de notre base de données (dump SQL), soit via nos outils de consultation.

2 Analyse du Sujet

2.1 Devinettes

Comme nous l'avons déjà mentionné, notre sujet consistait à développer un oracle lexical, i.e. un logiciel (qui a finalement pris la forme d'un site web) capable de "deviner" un mot auquel penserait l'utilisateur, dans le but final de permettre à un utilisateur de retrouver un mot qu'il aurait oublié mais qu'il serait capable de décrire.

Bien sûr, nous avons développé majoritairement l'oracle sous sa forme ludique, celle-ci étant plus efficace pour obtenir des données (certaines personnes peuvent passer jusqu'à plusieurs heures sur un jeu, alors qu'on ne cherche à retrouver un mot oublié qu'assez rarement).

Ce jeu s'est développé, comme nous l'avons expliqué plus tôt, sous la forme d'un mélange entre le site web Akinator pour la forme, et le jeu télévisé Pyramide pour le fond, et c'est donc tout naturellement que les interactions entre l'utilisateur et la machine se font sous la forme d'une succession d'indices fournis par l'un et de propositions de la part de l'autre, dans le but de deviner et faire deviner un mot choisi au préalable.

2.2 Contraintes de développement

2.2.1 Obtention des données

Conceptuellement, l'obtention de données lexicales se fait essentiellement par l'apprentissage de l'oracle. Les joueurs utilisant l'oracle peuvent le faire échouer à deviner un mot, ils doivent dans ce cas lui dire à quel mot ils pensaient. L'oracle doit donc créer de nouvelles relations (ou confirmer certaines relations déjà existantes) entre les indices donnés par le joueur et le mot à deviner.

2.2.2 Fiabilité et intégrité des données

Un risque évident concernant l'aspect "jeu" de l'oracle est de récolter des données peu précises voire même erronées. Pour s'assurer de la validité de ces données, une partie d'oracle faite par un joueur est retenue par notre système et ensuite posée plusieurs fois à d'autres joueurs.

Si des joueurs arrivent à deviner un mot qui a été proposé lors d'une devinette à l'oracle, différents liens entre les mots sont renforcés, nous permettant ainsi de juger de la pertinence et de la qualité de nos données.

Plaçons nous ici dans une situation où une devinette posée à l'oracle est proposée à un joueur. A chaque indice proposé au joueur, il fait une proposition. Le fait de trouver le mot à deviner déclenche les mécanismes de contrôle de fiabilité. Les liens suivants voient leur pondération augmenter :

- Le lien entre chaque indice proposé par le jeu et chaque proposition du joueur
- Le lien entre chaque indice proposé par le jeu et le mot à trouver

La première augmentation de pondération n'est pas triviale. Nous avons fait le choix d'augmenter de façon plus importante les liens entre les premiers indices et les premières propositions. En effet, la pertinence du lien entre le septième indice donné (par exemple) et la septième proposition du joueur est plus que douteux. Par contre le lien entre le premier indice et la première proposition est assez fiable, puisque l'esprit du joueur est uniquement concentré sur le premier indice, il proposera un mot proche de ce premier indice. Ensuite, ses propositions seront le résultat d'une association d'idées entre tous les indices.

La deuxième augmentation de pondération est plus simple. Le lien entre les indices (donnés à la base par un joueur dans une partie d'oracle) et le mot à trouver est assez certain dans le cas d'une réussite de la partie.

2.2.3 L'attrait du jeu

Plusieurs aspects ludiques ont été élaborés pour motiver les joueurs à donner des données de bonne qualité et en grande quantité.

Le principe de points, présent dans JeuxDeMots, permet au joueur de se positionner dans le jeu, de voir sa progression, de se comparer aux autres joueurs. On gagne des points de plusieurs façons dans Guess It ! :

- En effectuant une partie contre l'oracle, on gagne plus si l'oracle ne trouve pas
- En réussissant à trouver un mot auquel l'ordinateur pense

– A chaque fois qu'une devinette que l'on a proposé est trouvée par un autre joueur. C'est un bonus pour avoir proposé une partie contenant des données fiables

De plus, les joueurs peuvent avoir des amis dans le jeu et ainsi accéder à un classement par rapport à ceux-ci. C'est un facteur plus stimulant que la comparaison avec le score d'inconnus.

Enfin, dans le jeu où le joueur doit deviner un mot, le système donne des indices plus précis quand il le peut. Par exemple, si le joueur propose un mot synonyme de la réponse, cette indication lui sera fournie et pourra ainsi le motiver à continuer sa partie.

3 Outils utilisés

3.1 Langages et outils de développement

3.1.1 Langages de programmation

Nous avons décidé d'utiliser le langage PHP en ce qui concerne la génération de pages web pour des raisons pratiques. Notamment, JeuxDeMots a été développé en PHP et il est aisé d'installer le support PHP sur un serveur web.

De plus, nous avons utilisé le langage Python pour récupérer les relations de la base de données lexicales de JeuxDeMots. Ce langage, de part sa syntaxe simple à utiliser et sa puissance, nous a permis d'extraire la presque totalité de la base, tout en nous permettant de filtrer certaines relations, en un temps tout à fait honorable.

3.1.2 Bases de données

Nous avons de plus utilisé le SGBD MySQL, par commodité et parce qu'il est très répandu, simple à mettre en place et propose des performances correctes pour les tailles de données que nous avons à manipuler.

Cela dit, notre base est compatible SQL et donc implantable dans d'autres SGBD SQL tels que PostgreSQL, Oracle...

3.2 Outils de travail collaboratif

Tout au long du projet, nous avons été confrontés aux difficultés du travail en groupe. Pour tenter de pallier à ce problème, nous disposions notamment de deux outils :

3.2.1 Un gestionnaire de versions : Mercurial

Tout d'abord, nous disposions du gestionnaire de versions Mercurial (aussi appelé Hg), semblable à Git ou SVN, mais que nous avons préféré à ces deux alternatives car nous estimions qu'il offrait un meilleur compromis entre la puissance des fonctionnalités disponibles et la simplicité d'utilisation de l'outil.

Mercurial nous a permis, tout au long du semestre, de travailler en parallèle en synchronisant les données, fournissant un versionnement qui permet tout retour en arrière tout en laissant libre les modifications de chacun, même en cas de conflit.

De plus, l'arbre de visualisation créé au fil des différentes sessions de travail de chacun nous a permis de visualiser efficacement notre avancement tout au long de la réalisation du projet.

3.2.2 Coordonnement et attribution des tâches

Pour coordonner notre travail, nous avons prévu d'utiliser RedMine, un outil d'une grande puissance et aux fonctionnalités multiples.

RedMine propose de nombreux outils, comme par exemple un wiki, ou encore la possibilité de générer des taches (associées à des tickets) et de les attribuer aux différents membres du groupe de travail, ce qui permet d'éviter les "collisions" et le travail inutile.

Malheureusement, en raison des différents contre-temps auxquels nous avons fait face, nous n'avons pas pu profiter pleinement de la puissance de cet outil, ce qui nous a permis d'affronter les aléas du travail de groupe en situation d'urgence (cet aspect sera détaillé dans la partie 7.2 page 35).

3.3 Divers

Bien sûr, nous avons aussi utilisé des outils plus classiques, tels que l'échange de mails pour communiquer entre nous et avec notre encadrant. Nous pouvons aussi citer LaTeX, avec lequel nous avons rédigé notre rapport ainsi que les slides de notre présentation, ou bien encore Dia avec lequel nous avons dessiné les schémas de la base de données.

C'est avec GantProject que nous avons réalisé notre diagramme de Gantt lors de la rédaction du cahier des charges.

4 Modélisation

Dans tout projet de taille conséquente, la phase la plus importante, et aussi une des plus déterminantes quant à la forme finale du produit, est la phase de modélisation.

C'est au cours de cette phase que les concepts sont formalisés, les méthodes clairement définies et que les choix qui guident le développement sont faits.

Bien sûr, cette phase n'a pas été négligée, nous y avons même prêté une attention toute particulière.

4.1 Représentation des relations et associations

Le tout premier élément que nous avons eu à modéliser a été les relations entre les différents mots de la base de données. En effet, cet élément est la base du fonctionnement de l'oracle dans la mesure où c'est à partir de ces relations que tous les calculs pour deviner les mots sont faits.

Il a donc été crucial de trouver un moyen de les représenter qui soit simple, clair, intuitif, mais aussi efficace, tant en terme de vitesse de calcul que de compatibilité avec JeuxDeMots, dans la mesure où nos données en sont issues.

En terme de représentation abstraite, de simples liens entre mots nous a semblé être une bonne solution pour sa simplicité d'implémentation.

Ensuite, il a fallu décider de la manière dont allaient être représentées en machine ces données abstraites. Celles-ci devant être stockées dans une base de données, plusieurs choix s'offraient à nous.

Celui que nous avons retenu a été de représenter les relations entre les mots sous la forme d'une table d'association, un mot pouvant être relié à plusieurs autres. Cette méthode, en plus d'être proche de la représentation abstraite en terme de conceptualisation, permet des calculs nécessitant la bi-directionnalité des relations. Nous avons également décidé de représenter la force du lien entre deux mots par un poids attribué à chaque association.

En conclusion, tout au long de la phase de modélisation, nous nous sommes efforcés de respecter un seul mot d'ordre : la simplicité, afin d'augmenter la clarté et de faciliter au mieux la phase de développement.

Ci-après sont présentés les différents diagrammes représentant la modélisation des données dans la base et le déroulement d'une partie standard.

4.2 Tables utiles au jeu

En machine, la représentation des parties se fait grâce à trois tables : *game_ oracle*, *game_ oracle_ entry* et *game_ enigma*.

La première représente, comme son nom l'indique, une partie d'oracle jouée par un joueur. Elle se compose des attributs suivant :

- *id* qui est l'identifiant numérique et la clé de la partie,
- *word* étant le mot-réponse qui doit être trouvé par l'oracle. Ce champ est initialisé à "empty" et est mis à jour à la fin de la partie,
- *locked* qui indique si la partie est verrouillée en création ou en jeu,
- *locked_ date* qui permet de connaître la dernière fois où la partie a été verrouillée. Cela permet de savoir quelles parties peuvent être supprimées ou déverrouillées,
- *game_ owner* étant le pseudonyme du joueur ayant créé cette partie,
- *fail_ chip* et *succeed_ chip* qui sont les compteurs indiquant le nombre de fois où la partie peut encore être jouée.

Ces derniers attributs sont plus détaillés dans la section 5.1.1 page 19.

La deuxième table, *game_ oracle_ entry*, stocke quant à elle tous les indices donnés par le joueur lors de la partie oracle. Elle se compose de :

- *game_ id* étant l'identifiant de la partie oracle à laquelle l'indice correspond. C'est en fait une clé étrangère référençant le champ *id* de la table *game_ oracle*,
- *hint_ number* qui est le numéro de l'indice dans la partie. La clé de cette table est d'ailleurs le couple *(game_ id, hint_ number)*,
- *entry* étant l'indice en lui-même.

Les entrées de ces tables (*game_ oracle* et *game_ oracle_ entry*) sont supprimées au fur et à mesure qu'elles sont jouées (en tenant compte des jetons).

Enfin, la table *game_ enigma* stocke l'ensemble des parties enigma et enigma2 qui ont été jouées. Elle est composée des champs :

- *game_ id* l'identifiant numérique,
- *answer* le mot à trouver par le joueur,
- *locked* qui indique si la partie est verrouillée,
- *game_ owner* étant le pseudonyme du joueur ayant créé cette partie.

Il est à noter que cette dernière table n'est pas vraiment utilisée pour le moment, nous l'avons mise en place dans le cas où il serait nécessaire de faire des traitements spécifiques sur les parties enigma dans le futur.

4.3 Schémas

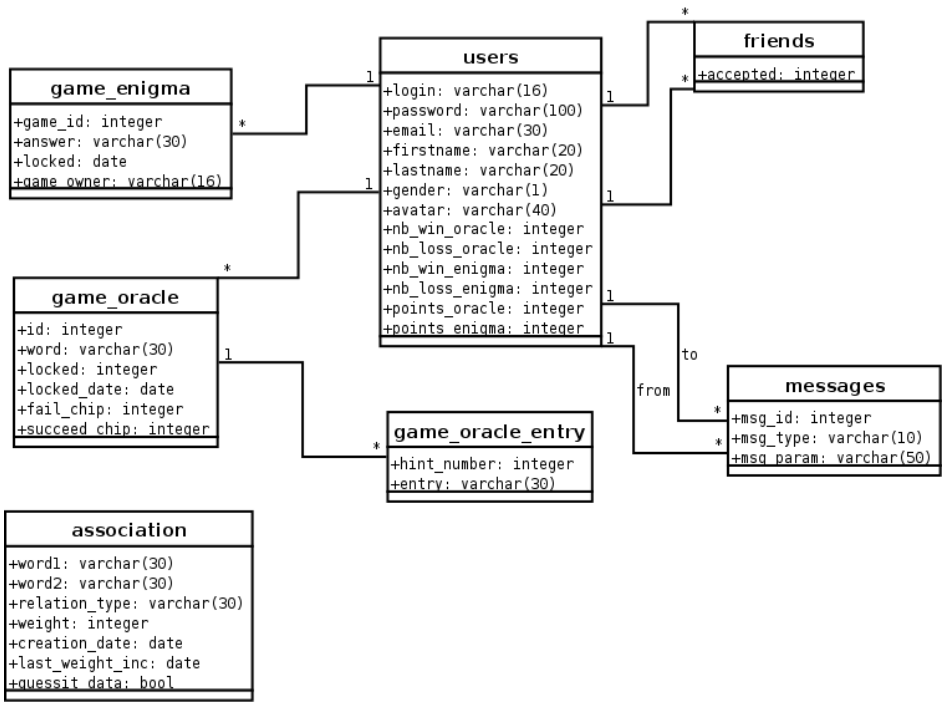

FIG. 4 – Schéma UML de notre base

4.4 Formalisation du calcul des mots

Soit $E = \{m_1, m_2, .., m_k\}$ les k premiers mots proposés en indice par un utilisateur de l'oracle (aussi bien le jeu que l'outil), on a alors V l'ensemble

des mots possibles à partir de E définit par :

$$V = \bigcap_{i=1}^{k} v(m_i)$$

Où les $v(m_i)$ sont, pour chaque mot m_i, l'union entre l'ensemble des mots sémantiquement associés à m_i et les mots tels que m_i appartient à leur ensemble de mots sémantiquement associés. Plus simplement, il s'agit des mots qui ont une relation avec m_i, qu'elle soit entrante ou sortante.
De plus on a :

$$P_i = \sqrt[|v(m_i)|]{\prod_{m \in v(m_i)} p_m}, \ \forall i \in V$$

Où P_i désigne le poids associé au i^{eme} mot de V en fonction des indices proposés, p_m le poids de la relation entre le i^{eme} mot de V et le mot m.
Le poids associé à un mot de V pour une série d'indices donnée est donc la moyenne géométrique des poids des relations entre ce mot et les différents indices proposés.

5 Développement

5.1 Partie Jeu

5.1.1 L'oracle

Comme on peut se l'imaginer, une partie d'oracle se déroule de façon assez simple. Lorsque la partie commence, le joueur doit choisir un mot (dans le cas de l'outil, c'est le mot dont l'utilisateur ne se souvient pas) puis donne à l'oracle les indices liés à ce mot un par un. L'oracle fait une proposition de réponse pour chaque indice donné jusqu'à ce qu'il échoue ou que l'utilisateur lui dise qu'il a trouvé.

La mécanique interne est assez simple : l'oracle sélectionne dans sa base tous les mots en relation avec l'indice du joueur (l'oracle utilise autant les relations où l'indice est en partie gauche que les relations où l'indice est en partie droite, ceci permettant d'obtenir le plus de mots possibles), puis choisit celui ayant la relation avec le poids le plus fort et supprime ce mot des propositions possibles dans la suite du jeu.
Dans le cas où l'indice est relié deux fois au même mot (c'est à dire une relation où l'indice est en partie droite et une relation où l'indice est en partie gauche) l'oracle ne prend en compte que la relation ayant le poids le plus fort.

Si le mot proposé par l'oracle n'est pas le bon, le joueur doit donner un autre indice ; l'oracle fait alors une nouvelle fois une sélection dans la base de tous les mots en relation avec ce nouvel indice.
L'oracle fait alors une intersection entre les mots sélectionnés grâce au premier indice et les mots sélectionnés grâce au second, ne donnant ainsi que les mots en relation avec les deux indices. Parmi ceux-là, l'oracle propose une nouvelle fois le mot ayant le plus fort poids, après avoir calculé la moyenne géométrique, pour chacun de ces mots, entre le poids de la relation vers le premier indice et le poids de la relation vers le second.

Le procédé se poursuit ainsi jusqu'à ce que l'oracle trouve le mot ou qu'il échoue, c'est-à-dire qu'il n'ait plus de mot à proposer (l'ensemble des mots se restreignant le plus souvent à chaque tour grâce à l'intersection) ou bien qu'il ait dépassé le nombre maximum de tours.
Il est à noter qu'à chaque tour, l'oracle effectue quelques traitements en vue d'éviter de proposer les mots que le joueur à déjà donné comme indice ou encore de ne pas soumettre une nouvelle fois un mot déjà proposé (si celui-ci avait encore la plus forte relation).

Dans le cas où l'oracle trouve le mot, le joueur se voit gratifié de points correspondant au nombre de tours dont l'oracle a eu besoin pour réussir.
Dans le cas où l'oracle échoue, le joueur doit donner le mot qu'il voulait faire deviner et se voit offrir de la même façon des points en fonction du nombre de tours plus un bonus pour avoir mis en déroute l'oracle (ce bonus peut d'ailleurs être facilement modifié grâce à un fichier de configuration).
Bien entendu, les points ne sont donnés au joueur que dans le cas où celui-ci est connecté à son compte, les anonymes ne pouvant logiquement gagner aucun point.

Nous avons également inclus une sécurité pour éviter une certaine forme d'anti-jeu ; en effet, il se peut qu'un joueur malveillant (ou alors très inattentif...) ne notifie pas à l'oracle que la dernière proposition de réponse était effectivement le mot à trouver, mais continue à jouer jusqu'à faire échouer l'oracle pour ensuite lui donner ce mot comme réponse. Ceci aurait pour effet de faire gagner, à chaque fois, un maximum de points au joueur.
Ainsi, l'oracle inspecte l'ensemble des mots qu'il a proposé au joueur et s'il trouve la réponse, il ne donne aucun point au joueur qui ne pourra de plus gagner aucun bonus grâce à cette partie ; ce malus est fait très simplement dans la base en passant cette partie sous la forme d'une partie faite par un joueur anonyme.

Quoiqu'il en soit, la partie que le joueur a créé est enregistrée dans la base en vue d'être proposée à d'autres joueurs dans la section enigma. C'est à ce moment que nous mettons en place le système de jetons.
Les jetons symbolises le nombre de fois qu'une partie peut être jouée dans le cadre du proposeur d'énigmes, ou, plus précisément, le nombre de fois où :

- une partie peut être réussie,
- une partie peut être échouée.

Ce système permet à la fois de ne pas surcharger le système de parties qui ne sont plus à jour, trop dures ou incohérente (grâce aux jetons d'échec) et de choisir le nombre de fois où une partie doit être réussie avant de ne plus être proposée (grâce aux jetons de réussites).
Il est à noter que le nombre de jetons est très facilement modifiable dans le fichier configuration.

5.1.2 Le proposeur d'énigmes

Le principe de jeu du proposeur d'énigmes, appelé plus simplement enigma, est également assez simple. Il se dédouble en deux fonctionnements aux finalités différentes : enigma qui utilise la base de données lexicales et enigma2 qui sert de validation de parties oracles.

La mécanique d'enigma est, dans le principe, assez simple dans le sens où la machine sélectionne un mot au hasard dans la base, que nous appellerons le mot à trouver, puis construit une liste d'indices en prenant aléatoirement dix mots en partie droite parmi les quarante plus fortes relations ayant en partie gauche le mot à trouver.
Lorsque le jeu commence, la machine donne le premier des indices au joueur, lequel tente de répondre un mot possiblement lié à lui.
Si la réponse du joueur n'est pas le mot à trouver, le proposeur d'énigme propose au joueur le deuxième mot de la liste d'indices et ainsi de suite jusqu'à ce que le joueur trouve ou qu'il n'y ait plus assez de tentatives possibles, synonyme d'échec du joueur.

L'attribution des points est simple : plus le joueur trouve rapidement le mot, plus il gagne de points et, bien entendu, s'il ne trouve pas, il ne gagne aucun point.
Par ailleurs, une victoire du joueur signifie également une augmentation des pondérations entre les indices donnés par la machine et les propositions du joueur (voir à ce sujet la partie 2.2.2 page 10).

Le principe d'enigma2 est sensiblement le même que celui d'enigma à ceci près qu'il utilise les parties oracles pour créer son énigme.
Il sert donc à la validation des relations amorcées pendant un oracle, ce qui fait de lui la partie essentielle de la dynamique d'expansion de la base de données lexicales.

En effet, le programme ne choisit pas ici un mot au hasard dans la base, mais une partie oracle, choisie aléatoirement. Il propose ainsi au joueur jouant la partie d'enigma2 les indications soumises par le joueur ayant fait la partie d'oracle.
Et, comme dans enigma, le jeu s'arrête lorsque le joueur trouve le mot ou qu'il ait dépassé le nombre de tentatives autorisées.

L'attribution de point est la même que celle de l'enigma premier du nom (c'est à dire plus le joueur gagne vite et plus il gagne de points) à la différence que la réussite d'une partie signifie ici un gain de points bonus pour le joueur

ayant créé la partie (si, bien entendu, il ne jouait pas en anonyme).

Le fonctionnement d'enigma2 est par nature plus compliqué que celui de son grand frère et plusieurs problèmes se sont posés :

- une partie d'oracle ne doit pouvoir être jouée en enigma2 par le même joueur,
- une partie d'oracle ne doit pouvoir être jouée avant la fin de sa création,
- une partie d'oracle ne doit pouvoir être jouée par deux joueurs en même temps.

Le premier point est essentiel à la mécanique de jeu ; en effet, si un joueur pouvait jouer ses propres parties, l'intégrité des relations en serait biaisée (en plus du fait que ce joueur gagnerait facilement des points). La solution à ce problème est assez simple : chaque partie oracle se voit attribuer le pseudonyme du joueur l'ayant créé ; ainsi, un joueur ne se verra proposer que des parties n'ayant pas le même pseudonyme.

Le deuxième problème est également très important pour le bon fonctionnement du jeu : lorsqu'une partie oracle est en train d'être créée (c'est à dire qu'un autre joueur essaye de faire deviner un mot à l'oracle), cette partie ne doit pas pouvoir être assignée à un joueur jouant à enigma2, notamment à cause du fait que le mot à trouver n'a pas encore été donné.
Ce problème est réglé par la mise en place d'un "flag", un attribut, mis à la valeur 1, signifiant que cette partie est en cours de création. Aucune partie ayant le "flag" à 1 ne peut être proposée aux joueurs. Ceci permet également de résorber le problème où le joueur quitte le jeu pendant sa partie d'oracle, impliquant que cette partie restera verrouillée.

Enfin, le troisième problème vient du fait qu'une partie est supprimée de la base une fois que l'un de ses compteurs de jetons arrive à zéro. Si deux joueurs jouent une partie n'ayant plus qu'un jeton de réussite, par exemple, et qu'ils réussissent tous les deux, il pourrait en résulter un problème d'intégrité de la base.
Ainsi, nous avons préféré jouer la carte de la sécurité en ne permettant pas à deux joueur de jouer la même partie en même temps ; ceci est fait très simplement en mettant la valeur 2 au "flag" vu ci-avant.

De façon sensiblement analogue à enigma, une victoire signifie une hausse des pondérations et/ou la création de nouvelles relations (voir également à ce sujet la partie 2.2.2 page 10).

Par ailleurs, comme cela a été évoqué dans la section 2.2.3 page 11, il faut noter que dans les deux enigma, le système peut proposer à chaque tour des indications aux joueurs.
S'il se trouve que le joueur propose un mot ayant une relation spécifique avec le mot à trouver, par exemple de synonymie ou encore de lieu, le système en avertira le joueur, l'aidant un peu et relançant, nous l'espérons, son engouement.

5.1.3 Interface

Plusieurs efforts ont été livrés dans la conception et dans le développement du site web qui sert d'interface entre l'oracle et les utilisateurs qui iront l'utiliser. Les discussions initiales autour de ce sujet nous ont fait comprendre qu'il était impératif que chacune des parties du site soit présentée de la façon la plus simple et rapide possible, pour que la compréhension par l'utilisateur soit quasi-immédiate indépendamment du fait que celui-ci soit jeune, adulte, ou plus âgé.

Pour cela, nous avons décidé de limiter au maximum la quantité de texte et d'informations affichées simultanément sur l'écran, en utilisant des phrases plutôt directes et concises, ainsi qu'en séparant le site en plusieurs pages différentes, chacune présentant une petite partie des informations concernant l'utilisateur et son compte dans notre jeu.

C'est dans cet esprit que nous avons préféré éviter au maximum de créer des pages avec des barres de défilement verticales, c'est à dire avec une taille plus grande que celle que le navigateur lui accorde. De cette façon, l'utilisateur possède tout ce dont il a besoin à l'instant dans son champ de vision directement, sans avoir à se déplacer dans la page pour effectuer la recherche d'une information ou donnée précise, ce qui finit par nous rapprocher sensiblement de l'esprit et du fonctionnement d'un jeu télévisé, comme Pyramide, cité auparavant dans ce rapport.

Un schéma de couleurs plutôt sobre a été choisi pour décorer notre site, composé basiquement de trois couleurs : le noir, le gris, et le bleu.

- Le noir, aussi utilisé pour la couleur du texte de notre barre personnelle, l'a été principalement en tant que couleur d'arrière-plan du site, ainsi que couleur de contour pour les plusieurs séparations internes à l'intérieur de chaque page du site.

– Le gris, quant à lui, nous a servi, dans une teinte claire, en tant que couleur de fond pour le cadre principal du site, tandis que nous avons utilisé une teinte de gris un peu plus foncée pour l'arrière-plan de la barre personnelle de l'utilisateur placée dans la partie inférieure de chaque page.

– Pour conclure, nous avons utilisé deux teintes différentes de bleu pour afficher l'ensemble du texte du site : tandis que le bleu marine a été utilisé pour les titres et sous-titres, nous avons choisi un bleu bien plus clair pour le reste du texte. Cette teinte, en plus de bien se mettre en valeur, se maintient dans un bon contraste avec le noir et les teintes de gris que nous avons choisi, et se mélange très bien avec le bleu marine, comme nous pouvons apercevoir sur la bannière principale présente sur chaque page du site.

FIG. 5 – Capture d'écran de la page d'accueil de Guess It !

Pour rester dans un esprit simple et pratique, tout en gardant un aspect décoré et plaisant, nous avons ajouté un ensemble d'images "cartoonesques"

représentant des personnages humains et machines, qui non seulement deviennent un peu les mascottes de notre jeu, mais aussi représentent bien l'idée de l'utilisation de l'oracle par un utilisateur humain. Ces images accompagnent l'utilisateur dans tout le jeu, et aident à maintenir un certain équilibre entre les données textuelles et graphiques présentes sur chaque page, comme nous pouvons l'observer sur la Figure 5 de la page 24, montrant la page initiale du site.

Le système de login/identification a été, quant à lui, conçu pour que l'utilisateur puisse bénéficier de toutes, ou du moins la plupart des informations qui lui sont nécessaires à n'importe quel moment, et surtout, sur n'importe quelle page du site, évitant ainsi qu'il ait à revenir sur sa page de profil utilisateur ou sur les différents classements proposés sur le site plusieurs fois pendant qu'il propose ou répond à des énigmes.

Pour cela, nous avons introduit une barre personnelle à chaque utilisateur, qui présente à l'utilisateur connecté une partie de ses informations personnelles, ainsi que des aperçus sur le classement général et son classement par rapport à ses amis, proprement affichés et instantanément mis à jour. Pour des informations ou données plus précises ou complètes, une série de liens vers différentes pages du site est affichée dans la partie inférieure de la barre. Pour éviter une trop grande concentration de texte sur celle-ci, nous avons rajouté l'image d'avatar de l'utilisateur sur la barre en tant qu'élément graphique, ce qui renforce cette idée de personnalisation envers l'utilisateur connecté sur le site.

Dans le cadre ludique de l'interface, un système d'avatars, des images représentant les utilisateurs sur le site, a aussi été mis en place. Plusieurs images sont proposées aux joueurs qui choisissent chacun un avatar auquel ils s'identifient le plus. Ces images, assez drôles pour la plupart, permettent d'ajouter un peu de couleur aux pages du site où les utilisateurs sont représentés, comme les classements, les pages de profil, ou encore la page de gestion des amis.

D'ailleurs, nous pouvons considérer que cet espace de gestion de ses amis sur le site est plutôt une bonne réussite pour la présence d'un certain dynamisme dans le site, étant donné qu'il est possible, de façon très simple d'inviter des gens à faire partie de son cercle d'amis ou d'annuler une invitation, de vérifier si on possède des invitations en cours et décider si on l'accepte ou pas, ainsi que d'obtenir des informations sur les membres de son cercle d'amis. Par ailleurs, les avatars de chacun des amis d'un utilisateur

sont utilisés pour générer une intéressante représentation graphique de son cercle d'amis.

Pour conclure au sujet de l'interface, nous pouvons aussi souligner le fait que nous avons, tout au long du projet, essayé de séparer au maximum la partie de programmation proprement dite, avec les états, actions, traitements de données et requêtes SQL, de la génération de code pour l'affichage de ces données traitées en langage HTML. De cette façon, nous avons pu, à de nombreuses reprises, avoir du code facilement compréhensible et ré-adaptable, mais aussi, avoir la possibilité de modifier la façon dont nous effectuons nos traitements sans que cela affecte le rendu graphique de nos pages.

5.2 Partie Outil de consultation

5.2.1 L'intérêt

LexiTool est la version "utilitaire" de l'oracle, c'est à dire la version destinée à aider un utilisateur ayant besoin de retrouver un mot (ou même l'orthographe d'un mot).

LexiTool suit exactement le même procédé que l'oracle (voir à ce sujet la section 5.1.1 page 19), c'est à dire, et pour faire simple, des sélections successives de mots suivies d'intersections jusqu'à ce que l'oracle indique au joueur le bon mot ou n'ait plus de proposition à faire.

La différence principale est que LexiTool ne fait pas partie de la "composante jeu", et ainsi ne permet ni de gagner des points, ni de créer de nouvelles parties destinées à être jouées dans enigma2.
En effet, ici, si l'oracle échoue, il présente juste ses plus plates excuses à l'utilisateur, et s'il réussit à trouver le mot, la simple gratitude de l'utilisateur suffira à emplir son coeur de bonheur.

LexiTool est donc un simple service donné aux utilisateurs.

5.3 Partie Administration

5.3.1 Administration des parties verrouillées

Comme vous l'aurez sans doute deviné, il nous a été obligatoire de trouver un moyen de traiter les parties verrouillées, c'est pourquoi nous avons introduit une page accessible uniquement aux administrateurs.
Le principe est simple, la page ne contient que deux boutons :

- le premier permettant la suppression de toutes les parties étant dans le premier état de verrouillage, c'est à dire les parties n'ayant pas été créées totalement lors d'une partie oracle
- le second permettant le déverrouillage de toutes les parties étant dans le deuxième état de verrouillage, c'est à dire les parties ayant été commencées par des joueurs dans enigma2 sans qu'elles soient terminées

Bien entendu, de tels traitements ne doivent pas toucher aux parties qui sont en train d'être créées ou en train d'être jouées, c'est pourquoi nous ne traitons que les parties verrouillées depuis plus de deux jours, temps très large ne laissant aucune chance de créer des problèmes dans la base.

Enfin, dans un soucis d'information, chacune de ces deux opérations indique à l'administrateur le nombre de parties supprimées ou déverrouillées.

5.3.2 Lexidata et Lexintersect

Lexidata et Lexintersect sont des outils de consultation de la base de données lexicales.
De part leur nature, ces pages sont bien évidemment accessibles uniquement aux administrateurs.

Lexidata permet de donner à l'administrateur qui l'interroge l'ensemble des relations d'un mot, autant celles où ce mot est en partie gauche que celles où il est en partie droite.
Lexidata permet d'observer les principales informations des relations, c'est à dire :

- le mot en partie gauche
- le mot en partie droite
- le poids de la relation
- le type de la relation

Les relations apparaissant peuvent être classées par ordre croissant ou décroissant pour chacune de ces quatre informations.

Lexintersect, quant à lui, permet de simuler le fonctionnement de l'oracle. Il est demandé à l'administrateur d'entrer deux mots au départ, Lexintersect faisant alors une sélection de tous les mots en relation avec la première proposition de l'administrateur et tous les mots en relation avec la seconde (en supprimant les éventuels doublons dans le cas où une des deux propositions -voire les deux- soit en relation deux fois avec le même mot en partie droite et gauche), puis fait une intersection, ne laissant ainsi apparaître que les mots en relation avec les deux insertions de l'administrateur.

Il est ensuite possible pour l'administrateur d'entrer une nouvelle proposition qui subit alors le même traitement, ne laissant apparaître que les mots en relation avec les trois insertions et ainsi de suite jusqu'à ce que l'ensemble des propositions de l'administrateur n'ait plus de relation en commun.

Ces deux outils nous ont permis de tester avec facilité le comportement du système et ont été, de ce fait, très utiles.

5.4 Réponse au sujet

Nous avons donc, comme nous venons de le montrer, développé un oracle lexical, sous la forme de l'outil LexiTool, a priori capable de deviner un mot oublié par l'utilisateur mais qu'il serait tout de même capable de décrire au moyen de mots sémantiquement liés.

Bien sûr, les performances de l'oracle sont intimement liées à la pertinence des données lexicales à sa disposition. C'est pourquoi nous avons développé, comme cela était aussi demandé, un outil permettant d'enrichir, tant en quantité qu'en qualité, la base de donnée lexicales dont nous disposons.

Cet outil, à savoir le couple de jeux Guess It ! / Enigma, est capable d'obtenir de nouvelles données lexicales fournies par les utilisateurs et de tester leur pertinence en les proposant à d'autres joueurs.

Nous avons donc, à première vue, répondu au sujet initialement proposé. Cependant, la réalité est plus complexe.

En effet, bien que ces outils aient été effectivement développés et implémentés, la question de leur efficacité demeure entière. Or, comme nous l'avons déjà mentionné, leur performance est très dépendante des données que nous possédons, et donc l'analyse de celles-ci est fondamentale pour améliorer la capacité de l'oracle à deviner les mots des utilisateurs et leur proposer des énigmes plus pertinentes.

L'analyse des données recueillies ainsi que leur pertinence sera étudiées ci-après.

6 Analyse des données recueillies

Une fois que les données ont commencé à nous parvenir, nous avons pu commencer le vrai travail d'analyse, afin d'évaluer la qualité des données et celle de notre algorithme de recherche de mot à partir des indices, ceci étant un point fondamental pour l'amélioration de la qualité des réponses fournies par l'oracle.

6.1 Évaluation des performances de l'oracle

Comme expliqué précédemment, le but de l'analyse des données lexicales recueillies est double :

- Permettre d'évaluer la pertinence des données fournies par les utilisateurs et leur comportement face à l'oracle,
- Permettre d'évaluer la qualité de l'algorithme de calcul des mots en fonction des indices fournis et éventuellement de l'améliorer.

Pour y parvenir, nous disposions des outils Lexidata et Lexintersect, présentés précédemment, ainsi que de l'accès direct à la base de données de Guess It ! / Enigma. Ceci nous a permis de consulter à volonté l'ensemble des parties crées par les différents utilisateurs et ainsi de pouvoir constater quasiment en temps réel l'évolution des relations entre les mots, de leur poids et des différents mots présents dans la base.

Bien sûr, avoir les données n'est pas suffisant pour pouvoir les analyser, il nous a aussi fallu comprendre à quoi elles correspondaient.

Plus concrètement, le problème principal auquel nous avons dû faire face concernant l'analyse des données lexicales recueillies fut l'évaluation des performances de l'oracle, tant sur la partie Guess It ! que sur la partie LexiTool.

La raison de cette difficulté est la suivante. Il y a en fait trois cas possibles lorsqu'un utilisateur tente de faire deviner un mot à l'oracle :

- L'utilisateur soumet des données cohérentes à l'oracle, qui trouve le mot de l'utilisateur,
- L'utilisateur soumet des données cohérentes à l'oracle, qui ne trouve pas le mot de l'utilisateur,
- L'utilisateur soumet des données incohérentes à l'oracle (que ce soit volontairement ou pas).

Le cas épineux est alors celui où l'oracle ne parvient pas à trouver le mot que l'utilisateur tente de lui faire deviner. En effet, il y a trois hypothèses différentes dans ce cas là :

- La base de données est incomplète
- L'algorithme n'était pas adapté pour ce cas précis
- L'utilisateur a fourni des données incohérentes

Il est dès lors très difficile de déterminer si, d'une manière générale, les performances de l'oracle sont mauvaises ou si ce sont les utilisateurs qui en compromettent le bon fonctionnement.

Partant de ce constat, nous avons cherché à déterminer une mesure de référence à laquelle nous pourrions comparer le taux de réussite de l'oracle.

L'idée qui nous est alors venue en premier lieu a été d'attribuer un score à l'oracle, de façon à ce qu'il puisse gagner des points, à l'image des humains. Le but aurait alors été de comparer le taux de réussite de l'oracle à celui des humains à travers le ratio des points obtenus sur le nombre de parties jouées. Cependant, par manque de temps, nous n'avons pu implémenter cette solution.

Cette absence de point de comparaison pour l'oracle a malheureusement fortement compromis l'analyse des données lexicales recueillies. Mais nous verrons par la suite que cela n'a pas été le seul écueil auquel nous nous sommes heurtés lors de l'analyse des données.

Nous allons à présent tenter de considérer les limites, en terme de pertinence, de cette analyse.

6.2 Pertinence

Comme nous l'avons déjà mentionné plusieurs fois, bien que la qualité des données recueillies soit bien entendu très importante, il est tout aussi fondamental d'en obtenir une quantité significative dans la mesure où l'étude que nous avons mené se basait sur une analyse statistique.

En effet, le but était, dans l'ensemble, de connaître les associations d'idées les plus répandues entre les mots. Celles-ci variant d'une personne à une autre, il est important d'obtenir beaucoup de données, et de beaucoup de personnes différentes, afin que les résultats soient cohérents avec la réalité et

que les anomalies statistiques ainsi que les informations erronées éventuellement enregistrées malgré les différents mécanismes défensifs mis en place soient masquées par l'abondance de données effectivement cohérentes et correctes.

Malheureusement, comme nous l'expliquerons plus en détail dans la partie 7.2 page 35 du rapport, le manque de temps nous a limité dans la quantité de données lexicales obtenues, nous privant en fin de compte de la possibilité de pouvoir faire une analyse de la performance des mécanismes que nous avions développés.

En outre, et comme nous l'avons expliqué précédemment, l'absence de point de repère par rapport auquel comparer les performances de l'oracle pour deviner un mot proposé par un humain nous a très fortement restreint dans notre capacité à l'évaluer.

Il n'est donc pas véritablement possible de conclure quant à la pertinence et la qualité des données que nous avons pu recueillir du fait de leur trop faible quantité, ni sur la pertinence de notre algorithme pour le calcul des mots à partir des indices.

7 Conclusion

7.1 Ensuite ...

Le plus difficile reste encore à faire. En effet, l'analyse sémantique de texte est un domaine en plein développement. N'ayant pas pu terminer l'oracle dans un délai suffisamment court, nous n'avons pas eu l'occasion d'effectuer de façon approfondie l'étude et l'analyse statistique des données recueillies.

De plus, nombre de fonctionnalités peuvent encore être ajoutées, comme par exemple certaines possibilités que nous n'avons pas pu exploiter totalement concernant l'utilisation du typage des relations entre les mots : par exemple si le devineur tente "beau" et que l'indice suivant est "substantif", le devineur tentera ensuite "beauté". Cette possibilité d'exploiter le typage de la relation entre les mots est d'ores et déjà présente dans la base données de JeuxDeMots, cependant nous ne l'avons pas utilisée dans ce cas de figure par manque de temps.

Par ailleurs, dans le fonctionnement actuel de notre oracle, les nouvelles relations que nous créons ne peuvent être que de type "association", ce qui est assez dommage.

D'autre part, nous n'offrons pas une sécurité totale contre l'anti-jeu, notamment dans le fait qu'un joueur puisse créer une partie oracle en anonyme et la jouer en connecté, il ne gagnera bien sûr pas de bonus mais connaîtra la réponse. Ce problème pourrait être réglé grâce au stockage de l'adresse IP du joueur, plutôt que son simple pseudonyme.

Néanmoins, un tel problème n'est pas si important car, premièrement, cette sécurité peut être contournée par un simple changement d'IP et, deuxièmement, nous pouvons espérer avoir une telle quantité de parties en attente que la probabilité de tomber sur sa propre partie en anonyme soit extrêmement faible.

Un autre point qu'il pourrait être intéressant d'ajouter au projet : la mise en place d'un timer permettant la purge des parties verrouillées automatiquement à la place des boutons décrits dans la partie 5.3.1 page 28.

L'ajout de certaines fonctionnalité permettant de faciliter l'analyse des données lexicales recueillies est aussi envisageable, comme par exemple le fait

de faire gagner des points à l'oracle de la même façon que les utilisateurs, comme nous l'avions suggéré précédemment.

Enfin, bien d'autres choses restent encore à faire bien sûr quant au développement de l'oracle. Une bonne promotion du site est par exemple indispensable à son développement, ainsi que le perfectionnement des méthodes de calcul de champs sémantiques par exemple.

7.2 Regrets

Malgré toute la volonté et les efforts que nous avons pu mettre dans ce projet, nous n'avons pas réussi à aller jusqu'au bout de ce que nous voulions faire. Il y a ainsi plusieurs points que nous n'avons pu aborder.

Tout d'abord, nous avions prévu de nous séparer de la base de donnée lexicale au bout d'un certain temps, pensant que nous aurions déjà assez de relations entre les mots pour proposer une alternative au fruit du travail de Monsieur Lafourcade, et permettant ainsi de créer, grâce à la participation des joueurs, une base prenant un chemin différent avec, peut être, l'apparition de groupements de mots totalement distincts.
Chose qui est à présent impossible car nous manquons de temps pour lancer le projet dans la "nature" (il est néanmoins possible de sélectionner dans notre base de données lexicales les relations ayant été créées grâce à l'oracle lexical, ce qui permettra, dans le futur, ce type de traitement).

Comme dit précédemment, nous n'avons pu réellement mettre à l'épreuve notre projet par manque de temps, ce qui nous a empêché de pouvoir observer les variations de pondérations des relations entre les mots au fur et à mesure des parties et, ainsi, ne nous a pas permis d'affiner les valeurs d'augmentations de poids des relations.
De façon analogue, il nous a été impossible d'avoir un véritable recul sur les affectations de points aux joueurs lors des parties.
Nous avons toutefois pensé à inclure une "contre-mesure" en la présence du fichier de configuration, permettant de modifier rapidement et aisément les valeurs des augmentations de points et de poids (il est par contre impossible de le faire rétroactivement).

De plus, nous avions initialement prévu de développer un client léger sous la forme d'une interface autonome en GTK+ qui puisse s'exécuter en dehors

d'un navigateur web.

Enfin, nous aurions voulu de même effectuer la partie concernant la fouille de texte, qui aurait permis de sensiblement augmenter le nombre de relations entre les mots et aurait aidé à créer une véritable auto-alimentation de la base, mais nous n'avons malheureusement pas eu le temps de nous pencher dessus.

7.3 Fonctionnement du groupe de travail

Après avoir composé, lors de l'analyse, les rôles suivant les préférences de chacun, notre groupe s'est scindé en sous-groupes aux buts précis, ayant bien entendu évolué au cours du développement :

- Bruno s'est occupé de toute la partie interface du site, mettant en place le design, les comptes joueurs, et la dynamique du site (profils, amis, classements, etc...)
- Remy s'est orienté vers la partie fouille de données, cependant, comme cela a été mentionné dans les regrets, peu d'avancées ont pu être faites sur ce point
- Julien et Pierre se sont, quant à eux, chargés de toute la mécanique interne avec, entre autres, la gestion des requêtes permettant les mises à jour des relations, la création des jeux et des outils nécessaires aux administrateurs, les mécanismes d'aide au joueur etc...

Nos sous-groupes se sont montrés très autonomes les uns des autres, permettant ainsi une progression qui n'était pas dépendante de celle des autres pour une large partie du projet. Avancer ainsi nous a permis de nous concentrer sur nos parties tout en nous impressionnant lors de nos réunions, ce qui nous motivait à avancer plus encore.

Les sous groupes ont tout de même dû se concerter pour les changements concernant les parties communes du projet telles que la structure de la base de données. Les points nécessitant une concertation furent peu nombreux et mineurs.

De plus, nous aurions sans doute dû mettre en place une planification des tâches plus stricte pour nous forcer à respecter le calendrier de développement qui nous avions défini au début du projet.

7.4 Pour finir

Cependant, ces erreurs de jeunesse sont loin d'avoir été inutiles, mais ont été extrêmement formatrices ; elles permettront, dans le futur, de savoir à quoi nous en tenir pour de tels projets et, plus généralement, sont une aubaine à saisir et à réfléchir dans notre optique de suivre un parcours orienté recherche.

Enfin, et malgré tous les problèmes cités précédemment, nous pensons avoir réalisé un logiciel tout à fait fonctionnel qui dépasse de loin nos attentes initiales.
En effet, nous avons conçu, grâce aux efforts de chacun, un programme que nous croyons utile au travail de Monsieur Lafourcade, c'est à dire à l'élaboration d'une base de données lexicales fiable en faisant participer le plus de personnes possible.

Cela fût pour nous l'occasion de voir la programmation sous un angle que nous ne connaissions pas, celui où nous devons trouver des mécaniques intéressantes et entraînant le joueur dans une dynamique de jeu, et de nous interroger sur la pertinence de la validité des liens entre les mots ou encore sur l'émergence des relations annexes que nous pouvons tirer d'une partie.

Le plaisir de travailler sur un projet ne prenant pas que le côté programmation en compte, l'ébullition d'idées et de remarques pertinentes, une certaine admiration pour les créations des autres, ainsi que la sensation de participer à l'amélioration d'un véritable travail de recherche ont rendu l'élaboration de cet oracle lexical très agréable, bien que parfois stressante, et ont été réellement instructifs pour chacun d'entre nous.

8 Bibliographie

- http ://www.akinator.com/aki_fr/
- http ://www.lirmm.fr/jeuxdemots/jdm-accueil.php
- http ://www.lirmm.fr/pticlic/pticlic.php
- http ://fr.wikipedia.org/wiki/Pyramide_(jeu_t%C3%A9l%C3%A9vis%C3%A9)
- Arbre des usages d'un terme dans un réseau lexical évolutif, Mathieu Lafourcade, Alain Joubert

9 Remerciements

Nous remercions Mathieu Lafourcade pour son soutien, ses encouragements face à nos résultats, ses conseils et sa bonne humeur.

Nous tenons aussi particulièrement à le remercier pour les nombreuses informations qu'il nous a fourni concernant les bases de données de JeuxDeMots et Pticlic, les pistes de réflexion pour le développement des différents modes de jeu de l'oracle et les diverses indications, déterminantes, sur Akinator et Pyramide.