Les améliorations que nous avons faites sur la structure de donnée se sont portées sur le conteneur utilisé par la classe dico.

Nous avons tout d’abord utilisé une std::map au lieu d’un std::vector afin de pouvoir associer a chaque liste de mot leur nombre de lettre en clé.  
Nous utilisons cette map car cela permet de n’allouer de l’espace pour une liste de mot que lorsqu’il y a un mot possédant ce nombre de lettres.  
Par exemple, pour stocker un mot possédant 25 lettres, il faut redimensionner le vecteur pour qu’il ait 26 places, alors qu’il faut simplement insérer a la clé 25 une liste de mot dans une map. Cela permet donc un gain d’espace en mémoire.

Ensuite, en valeur de la map, nous utilisons une std::unordered\_set car elle permet des temps de recherche de complexité O(1) en moyenne, et O(n) dans le pire des cas.  
Nous avons aussi trouvé le conteneur std::set ayant un temps de recherche de complexité O(log(n)), mais avons quand même préféré l’unordered\_set car nous n’avons pas besoin du tri effectué par le set lors d’une insertion.

Pour les modifications mineures, nous avons déplacé la fonction parseFile dans la classe dico afin qu’elle puisse directement insérer dans la structure de donnée, avons donc modifié les fonction membre de cette classe en conséquence, et avons adapté la fctMagique afin qu’elle ne dédouble pas les lettres.

Enfin, dans le programme, nous avons laissé le choix a l’utilisateur d’utiliser le scrabble français ou anglais, ainsi que son fichier de dictionnaire.