Actes de la 6e conférence conjointe Journées d'Études sur la Parole (JEP, 33e édition),
Traitement Automatique des Langues Naturelles (TALN, 27e édition),
Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 22e édition)
Nancy, France, 08-19 juin 2020

FlexEval, création de sites web légers pour des campagnes de tests perceptifs multimédias

Cédric Fayet[1] Alexis Blond[1] Grégoire Coulombel[1] Claude Simon[1]
Damien Lolive[1] Gwénolé Lecorvé[1] Jonathan Chevelu[1] Sébastien Le Maguer[2]
(1) Univ Rennes, CNRS, IRISA, Lannion, France
(2) ADAPT Centre, Sigmedia Lab, EE Engineering, Trinity College Dublin, Dublin, Ireland
claude.simon.1@univ-rennes1.fr, {damien.lolive, gwenole.lecorve, jonathan.chevelu}@irisa.fr, lemagues@tcd.ie

RÉSUMÉ

Nous présentons FlexEval, un outil de conception et déploiement de tests perceptifs multimédias sous la forme d'un site web léger. S'appuyant sur des technologies standards et ouvertes du web, notamment le framework Flask, FlexEval offre une grande souplesse de conception, des gages de pérennité, ainsi que le support de communautés actives d'utilisateurs. L'application est disponible en open-source via le dépôt Git https://gitlab.inria.fr/expression/tools/flexeval.

ABSTRACT

FlexEval, creation of light websites for multimedia perceptual test campaigns.

This paper presents FlexEval, an application which allows to design and deploy multimedia perceptual tests in the form of a light website. Using standard and open web technologies, especially the Flask framework, FlexEval offers great design flexibility, guarantees of sustainability, as well as support of active user communities. The application is open-source and available via the Git repository https://gitlab.inria.fr/expression/tools/flexeval.

MOTS-CLÉS : Tests perceptifs, multimedia, web.

KEYWORDS: Perceptual tests, multimedia, web.

1 Introduction

De nombreux domaines de recherche requièrent la récolte d'avis d'utilisateurs, notamment en traitement automatique des langues et de la parole. Par exemple, il peut s'agir pour les utilisateurs de se prononcer sur le naturel de signaux sonores produits par des systèmes de synthèse de parole, la validité des réponses d'un *chatbot*, l'expressivité d'avatars signant en langue des signes ou encore la pertinence de résultats d'un moteur de recherche. Cet article présente FlexEval, un outil permettant de créer et déployer de tels tests sans requérir de développements informatiques lourds ou de complexes étapes de configurations de la part du concepteur. Nous en donnons tout d'abord une vue d'ensemble (section 2), puis présentons un exemple simple d'un test perceptif (section 3). Nous détaillons ensuite des aspects liés à la flexibilité de FlexEval (section 4), puis les aspects juridiques (section 5).

2 Vue d'ensemble

FlexEval modélise un test perceptif de manière générique comme l'enchaînement de différentes scènes (*stages*) que le concepteur décide d'agencer selon son besoin. Les scènes disponibles sont

HAL : hal-02768500.

listées ci-dessous.

— **(Test)** La principale scène consiste, pour un utilisateur donné, en une succession d'*étapes* identiques lui présentant un ou plusieurs *échantillons* de données (textes, signaux sonores, vidéos...), une ou plusieurs questions et sollicitant ses réponses. Des modèles pour plusieurs grands types de base sont déjà définis (AB, MOS, MUSHRA...) et facilement adaptables.
— **(Authentification)** Un autre type de scène permet d'associer un identifiant à chaque utilisateur et de contrôler l'accès à une campagne en cours. Plusieurs variantes sont proposées : identification anonyme, identifiant spécifié par l'utilisateur, accès par invitation...
— **(Formulaire)** Il peut également être nécessaire d'adresser un questionnaire pour récolter des informations indépendantes de tout échantillon (par exemple, l'âge et la localité de l'utilisateur, ses commentaires en fin de test, sa connaissance *a priori* du domaine...).
— **(Page)** Enfin, il est également souvent utile de pouvoir passer par une scène explicative, ne nécessitant aucune action de l'utilisateur, typiquement un tutoriel. Ceci se fait sous la forme d'une simple page HTML.

Pour le concepteur, la mise en œuvre d'un test perceptif s'appuie sur la définition de l'agencement des scènes et de leurs éventuels paramètres, ainsi que sur la spécification de l'ensemble de données parmi lesquelles les échantillons seront tirés pendant le test. De son côté, l'application assure la gestion des identifiants, le brassage équilibré des données et la possibilité pour les utilisateurs de suspendre, puis reprendre le test comme ils veulent. Il en résulte un site web que le concepteur peut placer de manière autonome sur un serveur de son choix, qui, à l'issue de la campagne de test, produira un fichier de tous les résultats obtenus, sous la forme d'un fichier CSV ou d'une base de données SQLite. L'exportation de média est possible sous la forme de champ BLOB [1] au format binaire.

FlexEval est indépendant de logiciels tiers et facile à installer car il s'appuie sur des logiciels standard de développement web. Du côté serveur, l'application utilise le framework Flask [2], fondé sur Python, et le gestionnaire de base de données embarquées SQLite. Du côté client, les technologies utilisées sont HTML5, CSS et JavaScript (notamment JQuery). L'ensemble du code est récupérable à l'adresse `https://gitlab.inria.fr/expression/tools/flexeval`

3 Exemple de test perceptif

Prenons l'exemple d'un test dans lequel on souhaite comparer des phrases produites par 2 systèmes de synthèse de la parole, référencés comme les systèmes *hyp* et *ref*. Pour un testeur donné, nous souhaitons que la session de test débute par une authentification, suivie d'un questionnaire, puis 40 étapes d'un test de type MOS (*Mean Opinion Score*). À chaque étape, nous supposons vouloir présenter un texte et un échantillon de parole correspondant, puis obtenir deux notes entre 1 et 5 de la part du testeur, par exemple sur les degrés de spontanéité et d'intelligibilité du signal de parole. Pour un ensemble de textes donnés, il existe un échantillon de parole associé pour chaque système de synthèse testé. Les échantillons présentés au testeur viendront aléatoirement soit de l'un, soit de l'autre. Une fois les 40 étapes terminées, le test se conclut par une page de fin.

Pour lancer le site d'une campagne d'évaluation, FlexEval s'appuie sur un répertoire créé par le concepteur et qui contient toutes les informations utiles. L'arborescence de ce répertoire est illustré par la figure 1. Nous supposons que les données associées à chaque système sont listées au sein de fichiers CSV. Par exemple, nous créons deux fichiers `hyp.csv` et `ref.csv` qui, pour chaque ligne, liste un texte et le chemin vers le fichier WAV associé. Sur cette base, la figure 2 montre les deux fichiers de configuration qui permettent de spécifier à FlexEval la campagne d'évaluation. D'une part, la définition de l'enchaînement des scènes prend la forme du fichier `structure.json`. Les scènes

1. `https://developer.mozilla.org/fr/docs/Web/API/Blob`
2. `https://flask.palletsprojects.com/`

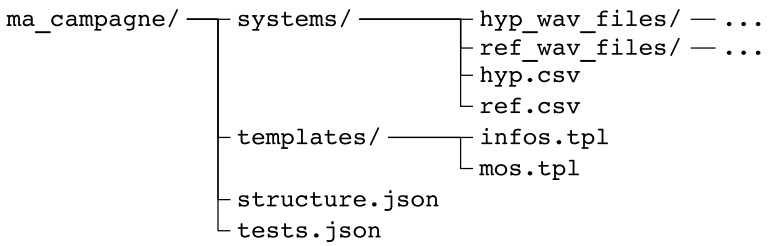

FIGURE 1 – Arborescence de fichiers de la campagne d'évaluation.

```
1 { "stages": {
2     "debut":{"type": "auth_by_invit", "next":"infos", ...},
3     "fin":   {"type": "page:user", ...},
4     "infos": {"type": "form", "template":..."next":"mos", ...},
5     "mos":  {"type": "test", "next": "fin",
6              "template": "mos.tpl", "nb_steps": 40, ...}
7 },
8 "entrypoint": "debut" }
```

structure.json

```
1 {
2   "mos":
3   [
4     {"name": "hyp", "data": "hyp.csv"},
5     {"name": "ref", "data": "ref.csv"}
6   ]
7 }
```

tests.json

FIGURE 2 – Configuration du scénario et des données du test perceptif.

sont identifiées par un nom et leur ordre de succession est indiqué par le champ `next`. Un squelette de page web peut être associée à chaque scène lorsque nécessaire. Par exemple, pour la phase `mos`, nous souhaitons utiliser un patron de fichier HTML (*template*) que nous avons défini : `mos.tpl`. Un exemple de rendu de ce patron est donné par la figure 3. Sans précision, un patron fourni par défaut est utilisé. D'autre part, la configuration pour l'accès aux données (fichiers CSV) de chaque système est définie dans `tests.json`. Les noms associés à chaque système seront ceux utilisés dans la base de données et au moment de l'export des résultats au format CSV. Ici, nous référençons nos systèmes *hyp* et *ref* et les associons à leur fichier CSV respectif.

Les différents fichiers mis en place, le concepteur lance le serveur web de FlexEval en indiquant le chemin vers le répertoire `ma_campagne/`, éventuellement en indiquant l'IP et le port auquel le site web sera accessible : `python3 run.py -ip 123.45.67.89 -port 8080 /chemin/vers/ma_campagne` . Le concepteur communique alors l'URL du site de la campagne. Une fois la campagne terminée, le concepteur récupère les réponses collectées via une interface d'administration dédiée ou directement à la racine du répertoire `macampagne/`. Le serveur web peut enfin être interrompu.

4 Éléments de flexibilité

FlexEval autorise une grande souplesse dans la création d'un test. Au niveau de la conception globale du test, l'emploi d'un fichier de configuration `structure.json` permet de décrire de manière flexible et intuitive un test sous la forme d'un scénario. Il n'y a pas de contrainte sur l'enchaînement des scènes. Si aucune phase d'authentification n'est prévue, l'outil en prévoira une par défaut, transparente pour l'utilisateur. Il est également possible de spécifier plusieurs scènes de tests successives.

Ensuite, le recours à des patrons de fichiers HTML et de possibles bibliothèques JavaScript permet une grande liberté d'interaction avec l'utilisateur que ce soit en terme de présentation des contenus (textes, vidéos, images…) que de récolte des réponses de l'utilisateur (multiples composants de formulaire, captation audio, vidéo…).

En outre, FlexEval permet l'inclusion de segments de code en Python au sein des patrons de fichiers

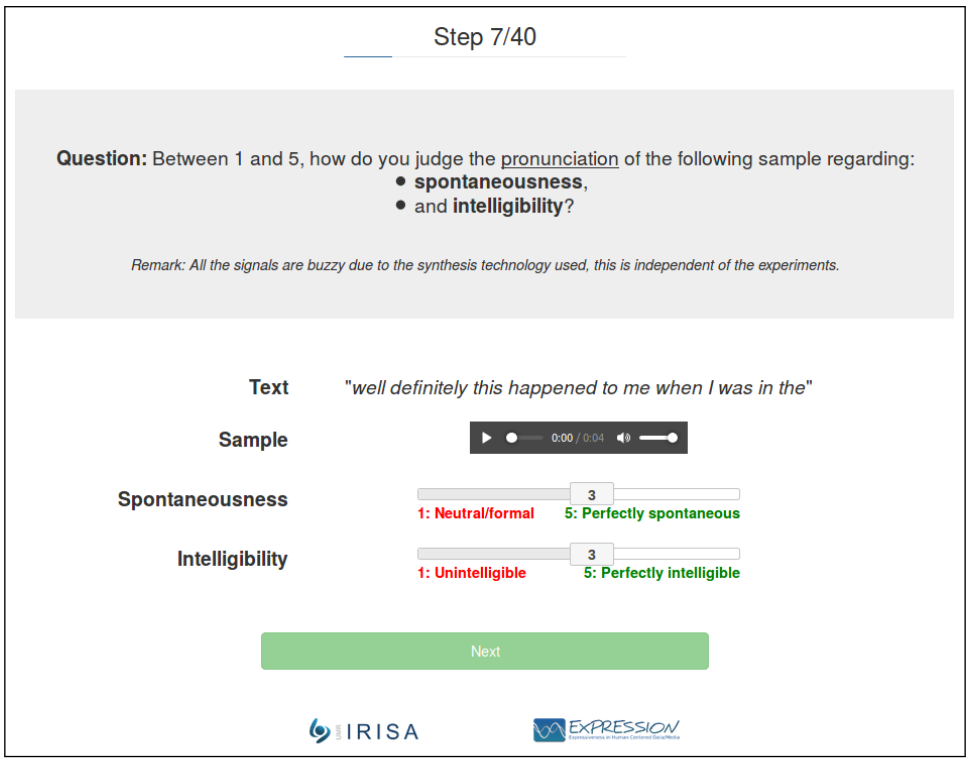

FIGURE 3 – Copie d'écran d'une étape d'un test MOS.

HTML, selon le langage de squelette Jinja [3]. Ces segments ont accès à des variables définies par FlexEval concernant l'utilisateur courant, la scène en cours (par exemple, pour une scène de test, le nombre d'étapes, le nombre de systèmes ou les informations sur un échantillon) ou encore des aspects explicitement fournis par le concepteur à travers le fichier `structure.json` (par exemple, le titre de la campagne, l'auteur des expériences...). Par ailleurs, les patrons peuvent inclure d'autres patrons, ce qui permet une meilleure réutilisation. Par exemple, il est possible d'inclure un pied de page dont le contenu (par exemple, des logos) est défini dans un autre fichier.

Enfin, il est aisé de créer ses propres types de scènes en Python en s'inspirant de celles existantes ou de définir des spécialisations de certains types par un mécanisme d'héritage, sans nécessité de modifier des fichiers existants de FlexEval.

5 Aspects juridiques

Tout responsable de l'utilisation de FlexEval au sein de son organisation doit se conformer aux obligations du Règlement Général sur la Protection des Données (RGPD [4]). Par exemple, au début du test, une page spécifiques doit présenter les principaux éléments liés à l'emploi de FlexEval mis en conformité avec le RGPD. Le consentement de l'utilisateur est rendu obligatoire à l'aide d'une case à cocher. Nous avons automatisé la génération du patron HTML de cette page spécifique en l'associant au document `legal.json`. Ce document est à compléter avec les informations exigées par le RGPD comme par exemple le nom et les coordonnées de l'organisation, les objectifs, la finalité et la nature des données collectées, les mesures de confidentialité, de droit d'accès, de sécurité ou de durée de conservation de ces données. On peut aussi y associer des données de clauses de CGU [5]).

3. http://jinja.pocoo.org/
4. https://www.economie.gouv.fr/particuliers/reglement-general-protection-des-donnees-rgpd
5. Conditions Générales d'Utilisation