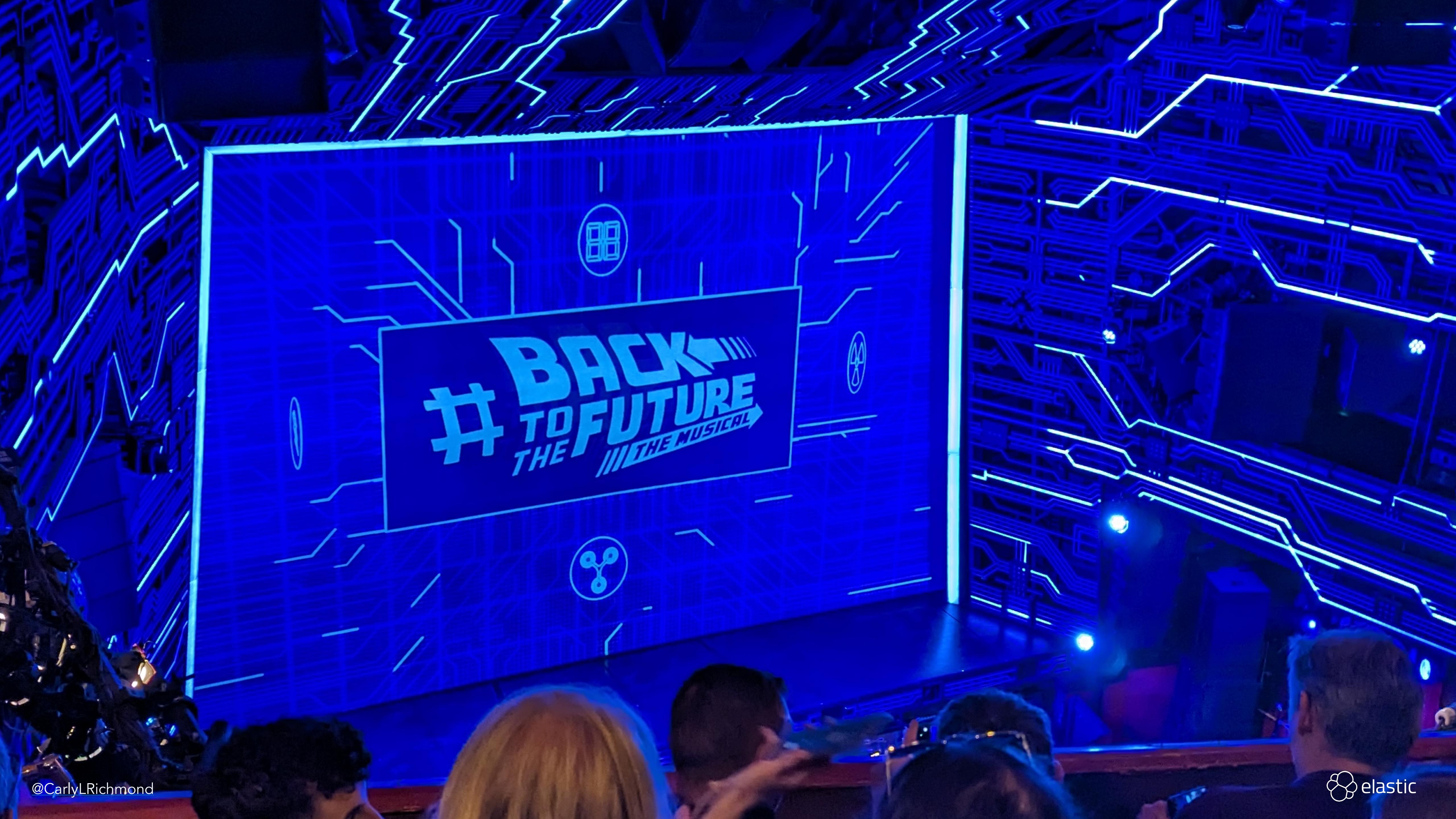


Revisited: Capturing Query Behaviour Using Elasticsearch

CARLY RICHMOND



2000





CAPTCHA



Type the word above

Verify

Luis von Ahn



TEDxCMU

GWAP



@CarlyLRichmond

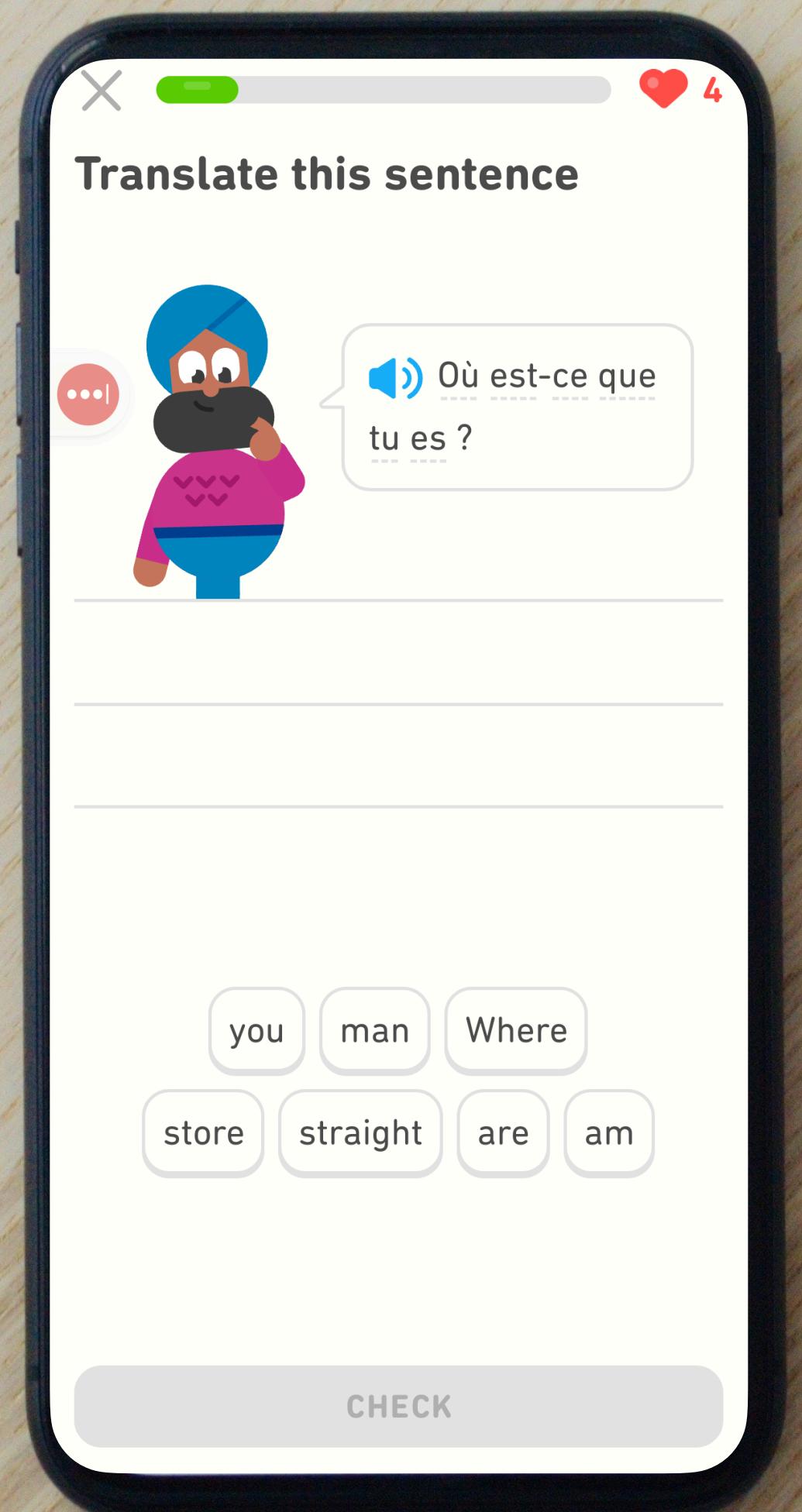


elastic

Game With A Purpose



Game With A Purpose



2011



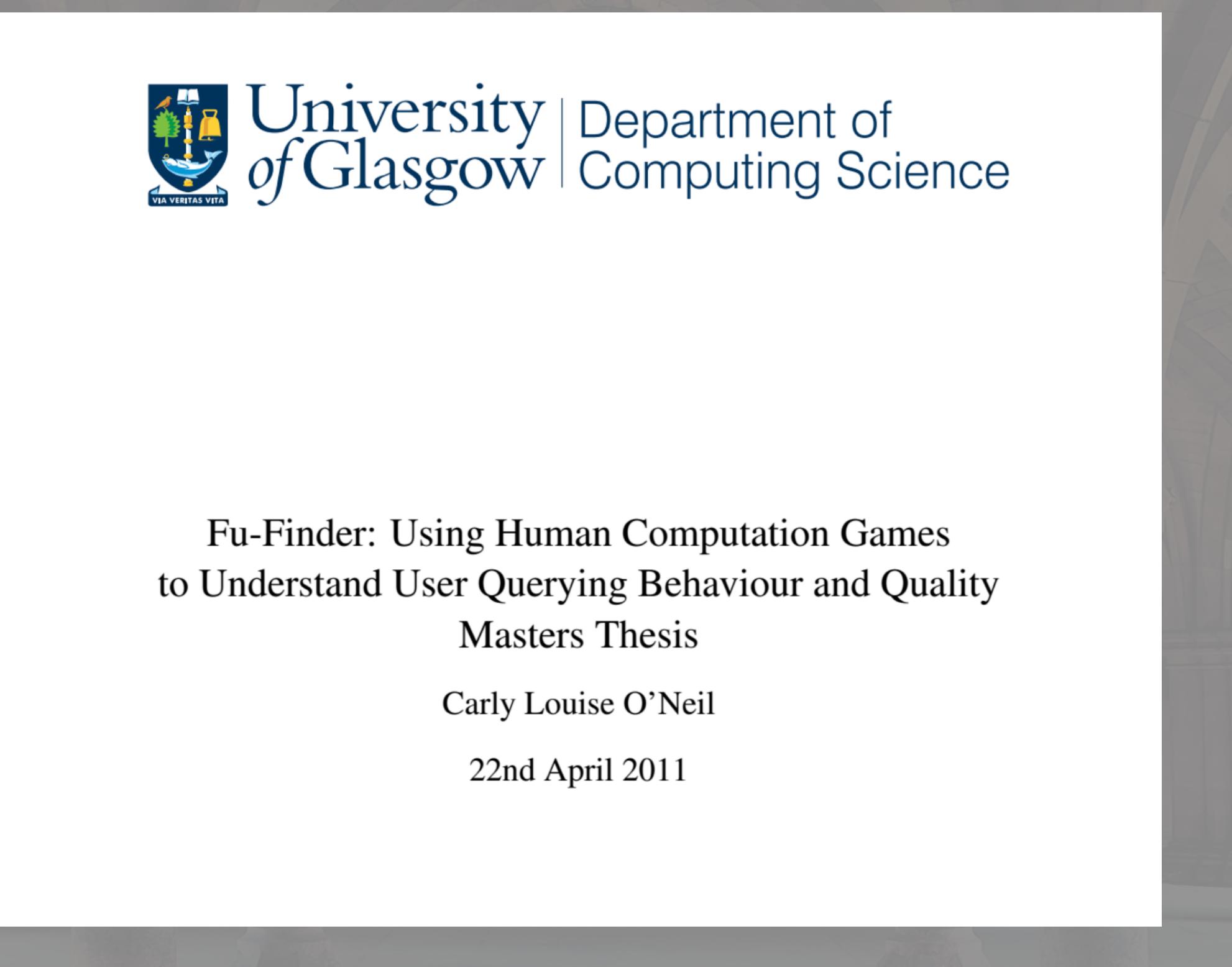


@CarlyLRichmond

elastic

The screenshot shows the Fu-Finder game interface. At the top, there's a small character icon and the text "Hello cawee! 😊". Below that is a timer showing "Time Remaining 1 : 32". The main area has a white background with a green border. It contains a search bar with the placeholder "Enter Query to Find the Given Page:" and a "Find" button. To the right of the search bar, it says "Total Score: 0" and "Query: enrique". Below that is a "Score: 900" and a "Next Page" button. A large red arrow points downwards from the search bar area towards the search results. The results are grouped into three columns labeled A, B, and C.

	A	B	C
1	Enrique Iglesias http://www.enriqueiglesias.com/	Enrique Iglesias http://www.enriqueiglesias.com/	Enrique Iglesias http://www.enriqueiglesias.com/
2	Enrique Iglesias - Wikipedia, the free encyclopedia http://en.wikipedia.org/wiki/Enrique_Iglesias	Enrique Iglesias - Wikipedia, the free encyclopedia http://en.wikipedia.org/wiki/Enrique_Iglesias	Enrique Iglesias - Wikipedia, the free encyclopedia http://en.wikipedia.org/wiki/Enrique_Iglesias
3	YouTube - Enrique Iglesias - I Like It http://www.youtube.com/watch?v=3Fv%3DX9_njA	Enrique: Information from Answers.com http://www.answers.com/topic/enrique	Enrique - Wikipedia, the free encyclopedia http://en.wikipedia.org/wiki/Enrique
4	Amazon.com: Enrique: Enrique Iglesias: Music http://www.amazon.com/Enrique-Iglesias/dp/B000030011	Enrique Iglesias: Information from Answers.com http://www.answers.com/topic/enrique-iglesias	Enrique Iglesias http://www.enriqueiglesias.com/main
5	Enrique Iglesias Free Music, Tour Dates, Photos, Videos http://www.myspace.com/enriqueiglesias	Enrique - Wikipedia, the free encyclopedia http://en.wikipedia.org/wiki/Enrique	Enrique Iglesias tickets, concerts and tour dates. Official ... http://www.ticketmaster.co.uk/Enrique-Iglesias-tickets/artist/777945



Fu-Finder: A Game for Studying Querying Behaviours

Test your Search-Fu

Carly O'Neil
School of Computing Science
University of Glasgow
United Kingdom
oneilc@dcs.gla.ac.uk

James Purvis
School of Computing Science
University of Glasgow
United Kingdom
0801303P@student.gla.ac.uk

Leif Azzopardi
School of Computing Science
University of Glasgow
United Kingdom
leif@dcs.gla.ac.uk

ABSTRACT

Usually the focus of evaluation within Information Retrieval has been placed largely upon the system. However, the individual user and their submitted queries are typically the greatest source of variation in the search process. This demonstration paper presents Fu-Finder, a fun and enjoyable game that measures the user's querying abilities (or search-fu). This game provides useful data for the study of user querying behaviour and assesses how well users can find specific web pages using different search engines.

Categories and Subject Descriptors: H.3.3 Information Storage and Retrieval: Information Search and Retrieval

General Terms: Performance, Experimentation

Keywords: Search-Fu, Querying Behaviour, Findability

1. INTRODUCTION

Searching and finding particular webpages can sometimes feel like a game of skill and chance. Choosing the "right" query terms to illicit the desired response is largely dependent upon the user. The user needs to be able to understand their information need, the system that they are using, how it works, what it is in the document (imagined, ideal, actual and/or known) and what terms would help to distinguish it from other documents. Not a particularly easy task. This has led to the term, *search-fu* being coined which denotes the skill of user's search abilities i.e., someone with weak search-fu will be unlikely to find relevant documents, while someone with strong search-fu will be able to find almost anything. So the question is, how much search-fu does a user possess? Or stated more formally, how well can a user satisfy their information needs when using a search engine?

The prototype system we have developed to help answer this question is a human computation game (HCG). HCGs have been developed to obtain data for various purposes (such as image annotation, character recognition, etc) [8]. Recently, the game Hunt was developed to help optimise the Bing search engine as a way to help annotate pages

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.
Copyright 2011 ACM 978-1-4503-0717-8/11/10 ...\$5.00.

that are difficult to find [3]. However, rather than trying to optimise a particular search engine, we have developed Fu-Finder to evaluate the user's ability at querying for pages across a number of search engines. The game is currently hosted and available to play¹, while the code is open source and freely available to download as part of the PuppyIR project housed on SourceForge². In this paper, we describe the game developed and some of the initial results.

2. EVALUATIONS USING GAMES

Human computation games provide a novel tool for solving computationally challenging problems (which are easy for humans, but hard for computers). The success of such games has been their ability to generate large quantities of reliable and useful data by providing users with enjoyment and points as payment rather than money or course credit. The added advantage is that these games can be undertaken in a controlled environment (i.e. within the remit of the game rules/setup) much like a standard or traditional IR experiment, but without the problems of using paid-workers through crowd sourcing. While HCGs are often developed to solve a difficult problem, such as von Ahn's ESP game, where players independently annotate images [7], in IR human computation games have been used for evaluation purposes.

Two notable games developed in IR are: Book Explorer [2] and PageHunt [3]. In [2], they developed a human computation game to assess relevancy of electronic books to topics. The main motivation for incorporating this task is a game was to provide a means of obtaining relevancy judgements for large document collections without needing to provide fiscal payment to study participants. This has been outlined as a prime motivation for including a computationally-hard task within a game with a purpose [7]. The Book Explorer game, whose model is a variant of the inversion-problem model [8], has two player roles, which can be adopted by individuals or teams. The *explorer* locates and highlights relevant sections of a document; while the *reviewer* provides relevancy feedback on documents [2]. Reviewers can only enter relevancy judgements for documents that have been evaluated by at least one explorer player. Furthermore, reviewers also serve as a cheating-detection mechanism as they examine explorer input, thus creating a multi-level validation system.

While the results from the study appear to show some

¹<http://www.dcs.gla.ac.uk/access/fufinder>

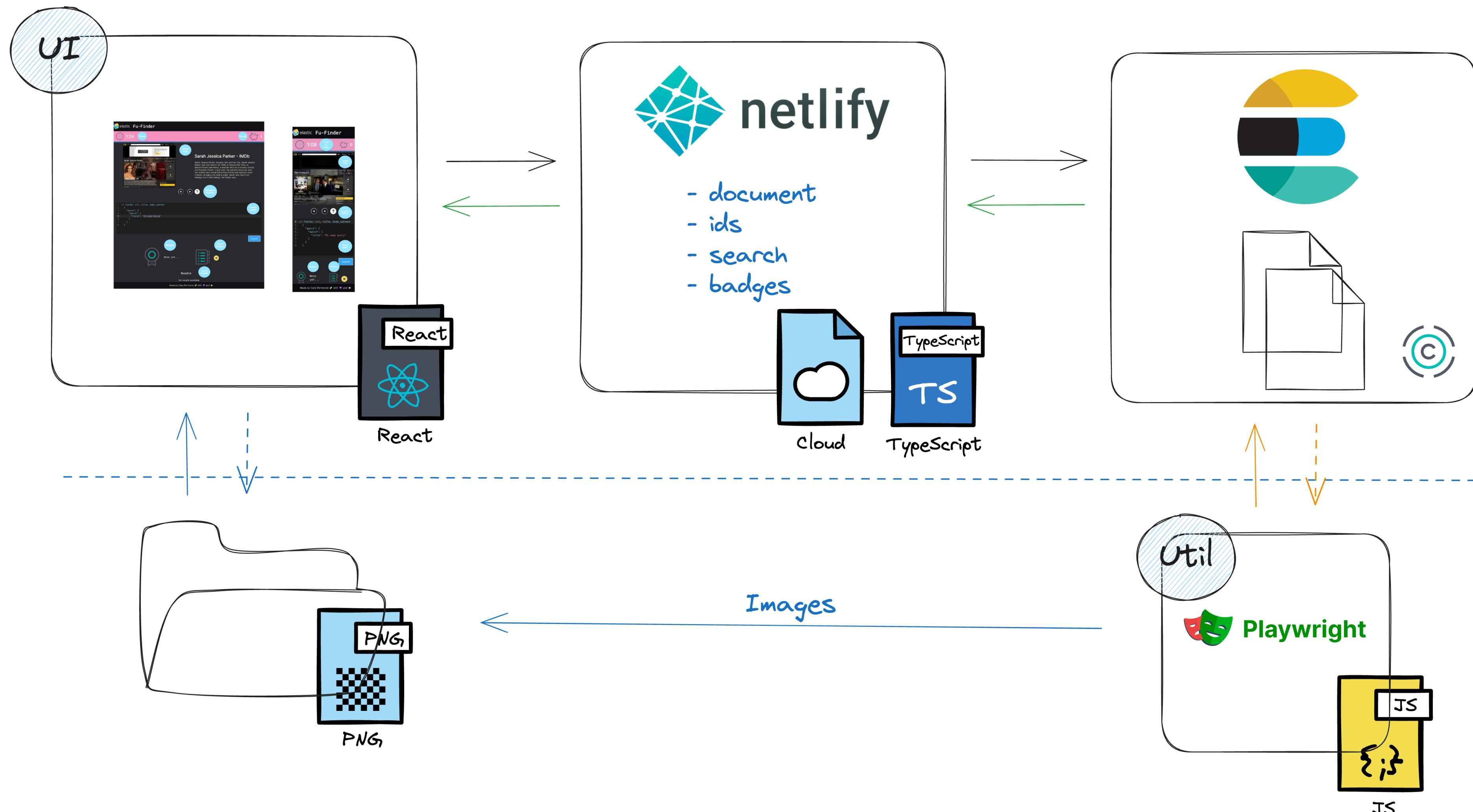
²<http://sourceforge.net/projects/puppyir/>

2023

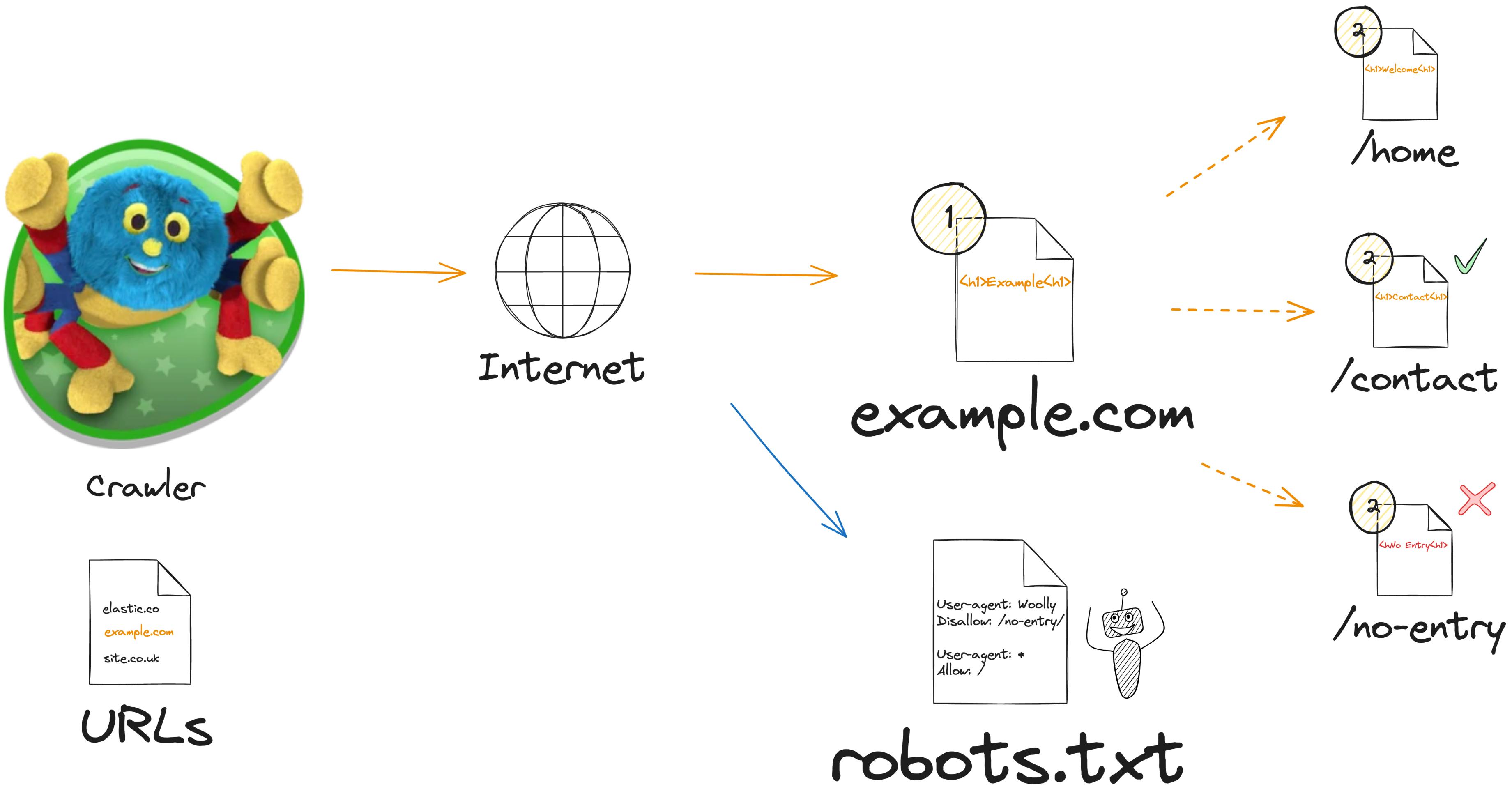




Architecture



Web Crawler



Elastic Web Crawler

The screenshot shows the Elastic Enterprise Search interface with a modal dialog titled "Custom crawl configuration". The dialog is used to set up a one-time crawl with custom settings. It includes fields for "Max crawl depth" (set to 1), "Add domains to your crawl" (listing 202 selected domains including coffeereview.com, imdb.com, baf.com, aromatherapy.com, walmart.com, filext.com, and others), and "Seed URLs" (listing 244 selected entry points including reviews from coffeereview.com, various IMDb URLs, and others). The background shows the "search-elastic-fu-finder-page" index overview with 202 documents and a crawler ingestion type.

Keyword Document

```
● ○ ● keyword-kate-winslet-imdb.json

1 {
2   "_index": "search-elastic-fu-finder-pages",
3   "_id": "63d3a0d21238d1dc003b9c79",
4   "_score": 4.4962697,
5   "_source": {
6     "last_crawled_at": "2023-01-27T14:46:47Z",
7     "additional_urls": [
8       "https://www.imdb.com/name/nm0000701/"
9     ],
10    "body_content": "Kate Elizabeth Winslet was born in Reading, Berkshire...",
11    "domains": [
12      "https://www.imdb.com"
13    ],
14    "title": "Kate Winslet - IMDb",
15    "meta_keywords": "Biography, Photos, Awards, News",
16    "url": "https://www.imdb.com/name/nm0000701/",
17    "url_scheme": "https",
18    "url_port": 443,
19    "url_host": "www.imdb.com",
20    "url_path_dir2": "nm0000701",
21    "url_path": "/name/nm0000701/",
22    "url_path_dir1": "name"
23  }
24 }
```

Setting up the Embedding Model

The screenshot shows the Hugging Face Model Hub website. On the left, there are filters for Tasks (Fill-Mask, Question Answering, Summarization, etc.), Libraries (PyTorch, TensorFlow, JAX), Datasets (wikipedia, common_voice, bookcorpus, glue, squad, dcleuroparl jrc-acquis, conll2003, oscar), Languages (en, es, fr, de, zh, sv, fi, ja), and Licenses (apache-2.0, mit, cc-by-4.0). A search bar at the top has the placeholder "Search models, datasets, users...". In the center, a list of models is displayed with "Models 27,459" and a filter "Sort: Most Downloads". One specific model is highlighted: "sentence-transformers/msmarco-MiniLM-L-12-v3" by "Helsinki-NLP/opus-mt-zh-en". The model details page shows its description, download count (1.5M), and a large blue button labeled "sentence-transformers/msmarco-MiniLM-L-12-v3". Below the main list, there are two other models: "sentence-transformers/bert-base-nli-mean-tokens" and "sentence-transformers/all-MiniLM-L6-v2".

```
$ eland_import_hub_model  
--cloud-id $ELASTIC_CLOUD_ID \  
--es-api-key $ELASTIC_API_KEY \  
--hub-model-id sentence-transformers/  
msmarco-MiniLM-L-12-v3 \  
--task-type text_embedding \  
--start
```

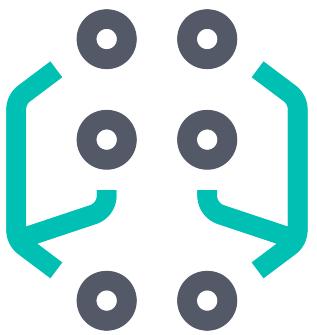
The screenshot shows the Elastic Machine Learning Model Management interface. On the left, there are sections for Machine Learning (Overview, Notifications, Memory Usage), Anomaly Detection (Jobs, Anomaly Explorer, Single Metric Viewer, Settings), Data Frame Analytics (Jobs, Results Explorer, Analytics Map), and Model Management (Trained Models, Data Visualizer, File, Data View, Explain Log Rate Spikes, Log Pattern Analysis). The "Trained Models" section lists three models: "elser_mod" (Elastic Learned Sparse Encoder v1), "lang_ident_model_1" (Model used for identifying input text), and "sentence-transformer_rs_msma_co-minilm_l-12-v3" (Model sentence-transformer-l-12-v3 for task type). The "Test trained model" section shows a test using the "sentence-transformers/msmarco-MiniLM-L-12-v3" model with the input text "Kate Elizabeth Winslet was born in Reading, Berkshire...". The "Text embedding" section displays the generated embeddings.



Select the appropriate model



Load the model to the cluster



Manage models

Generate Embeddings

The screenshot shows the 'Ingest Pipelines' section of the Elastic Stack Management interface. On the left, a sidebar lists various management sections like 'Management', 'Ingest Pipelines', 'Data', 'Alerts and Insights', and 'Security'. The main area displays a list of pipelines, with one pipeline named 'elastic-fu-finder-vector-pipeline' selected. This pipeline's configuration is shown in a modal window on the right, which includes a 'Description' field and a 'Processors' section containing JSON code for defining the pipeline's logic.

The screenshot shows the 'Dev Tools' section of the Elastic Stack Management interface, specifically the 'Console' tab. It displays a code editor with a POST _reindex command. The command specifies a 'source' index ('search-elastic-fu-finder-pages') and a 'dest' index ('vector-search-elastic-fu-finder-pages'). The 'dest' index is configured to use the 'elastic-fu-finder-vector-pipeline' for generating embeddings.

```
1 POST _reindex
2 {
3   "source": {
4     "index": "search-elastic-fu-finder-pages"
5   },
6   "dest": {
7     "index": "vector-search-elastic-fu-finder-pages",
8     "pipeline": "elastic-fu-finder-vector-pipeline"
9   }
10 }
```

The screenshot shows a terminal window displaying a JSON document titled 'vector-kate-winslet-imdb.json'. The document contains a single search result for 'Kate Winslet'. The result includes fields like '_index', '_id', '_score', '_source', 'text_embedding', and 'predicted_value', indicating the generated vector embedding for the query.

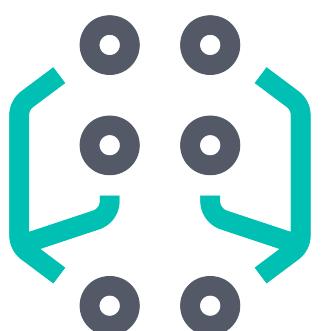
```
1 {
2   "_index": "vector-search-elastic-fu-finder-pages",
3   "_id": "63d3a0d21238dd1c03b9c79",
4   "_score": 4.4962697,
5   "_source": {
6     "last_crawled_at": "2023-01-27T14:46:47Z",
7     "text_embedding": {
8       "predicted_value": [
9         0.23861193656921387,
10        -0.3623441457748413,
11      ],
12      "is_truncated": true,
13      "model_id": "sentence-transformers__msmarco-minilm-l-12-v3"
14    },
15    "additional_urls": [
16      "https://www.imdb.com/name/nm0000701/"
17    ],
18    "body_content": "Kate Elizabeth Winslet was born in Reading, Berkshire...",
19    "domains": [
20      "https://www.imdb.com"
21    ],
22    "title": "Kate Winslet - IMDb",
23    "meta_keywords": "Biography, Photos, Awards, News",
24    "url": "https://www.imdb.com/name/nm0000701/",
25    "url_scheme": "https",
26    "url_port": 443,
27    "url_host": "www.imdb.com",
28    "url_path_dir2": "nm0000701",
29    "url_path": "/name/nm0000701/",
30    "url_path_dir1": "name"
31  }
32 }
```



Enrich data using
inference processor



Reindex data using
ingest pipeline



Generate
embeddings

Vector Document



vector-kate-winslet-imdb.json

```
1 {
2     "_index": "vector-search-elastic-fu-finder-pages",
3     "_id": "63d3a0d21238d1dc003b9c79",
4     "_score": 4.4962697,
5     "_source": {
6         "last_crawled_at": "2023-01-27T14:46:47Z",
7         "text_embedding": {
8             "predicted_value": [
9                 0.23861193656921387,
10                -0.3623441457748413,
11            ],
12            "is_truncated": true,
13            "model_id": "sentence-transformers--msmarco-minilm-l-12-v3"
14        },
15        "additional_urls": [
16            "https://www.imdb.com/name/nm0000701/"
17        ],
18        "body_content": "Kate Elizabeth Winslet was born in Reading, Berkshire...",
19        "domains": [
20            "https://www.imdb.com"
21        ],
22        "title": "Kate Winslet - IMDb",
23        "meta_keywords": "Biography, Photos, Awards, News",
24        "url": "https://www.imdb.com/name/nm0000701/",
25        "url_scheme": "https",
26        "url_port": 443,
27        "url_host": "www.imdb.com",
28        "url_path_dir2": "nm0000701",
29        "url_path": "/name/nm0000701/",
30        "url_path_dir1": "name"
31    }
32 }
```

Elasticsearch Client



elasticsearch.ts

```
1 const vectorSearchIndex = 'vector-search-elastic-fu-finder-pages';
2
3 const cloudID = process.env.ELASTIC_CLOUD_ID;
4 const apiKey = process.env.ELASTIC_API_KEY;
5
6 const client = new Client({
7   cloud: { id: cloudID },
8   auth: { apiKey: apiKey },
9 });
10
11 export async function getSearchResults(query: any) {
12   const keyword = query.query;
13   const vector = query.knn;
14
15   return client.search({
16     index: vectorSearchIndex,
17     profile: true,
18     query: keyword,
19     knn: vector
20   });
21 }
```

Hits Processing



get-results.ts

```
1 async function getResults(newQuery: string) {
2   setShowSpinner(true);
3
4   try {
5     const response = await axios.post('.netlify/functions/search', { queryString: newQuery });
6     const results = response.data.hits?.hits;
7     const queryTypes = response.data.profile?.shards[0].searches.flatMap((search: { query: any[]; }) => {
8       return search.query.map((element: { type: any; }) => {
9         return element.type;
10      })
11    });
12
13    // Check for match and calculate score here
14
15  }
16}
17 catch(error) {
18  console.log('Unable to get results');
19}
20 finally {
21  setShowSpinner(false);
22}
23}
```

Percolators

The screenshot shows the Elasticsearch Dev Tools Console interface. On the left, the history panel displays two requests:

- Line 246: PUT user-queries. This request defines a new index pattern named "user-queries". It includes mappings for properties like "message" (text type), "document_id" (keyword type), and a "query" field which is explicitly defined as a "percolator" type.
- Line 277: GET user-queries/_search. This request performs a search on the "user-queries" index. It uses a "percolate" query with the "query" field set to "susan". The "index" parameter is set to "vector-search-elastic-fu-finder-pages", and the "id" parameter is set to "63d3a0d21238d1060f3b9ca3".

On the right, the results panel shows the response to the search request. The response is a JSON object with the following structure:

```
1- {  
2-   "took": 19,  
3-   "timed_out": false,  
4-   "_shards": {  
5-     "total": 1,  
6-     "successful": 1,  
7-     "skipped": 0,  
8-     "failed": 0  
9-   },  
10-  "hits": {  
11-    "total": {  
12-      "value": 1,  
13-      "relation": "eq"  
14-    },  
15-    "max_score": 0.23289725,  
16-    "hits": [  
17-      {  
18-        "_index": "user-queries",  
19-        "_id": "JSFXuYsBmLtfKjaC8afi",  
20-        "_score": 0.23289725,  
21-        "_source": {  
22-          "document_id": "63d3a0d21238d1060f3b9ca3",  
23-          "query": {  
24-            "multi_match": {  
25-              "query": "susan",  
26-              "fields": [  
27-                "title",  
28-                "body_content"  
29-              ]  
30-            }  
31-          },  
32-          "knn": {  
33-            "field": "text_embedding.predicted_value",  
34-            "k": 10,  
35-            "num_candidates": 50,  
36-            "query_vector_builder": {  
37-              "text_embedding": {  
38-                "model_id": "sentence-transformers__msmarco-minilm-l-12-v3",  
39-                "model_text": "susan"  
40-              }  
41-            }  
42-          }  
43-        }  
44-      }  
45-    ]  
46-  }  
47-}
```

Console - Dev Tools - Elastic

Find apps, content, and more.

elastic

Dev Tools Console

Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Variables Help 200 - 0

```
246 PUT user-queries
247 {
248   "mappings": {
249     "properties": {
250       "message": {
251         "type": "text"
252       },
253       "document_id": {
254         "type": "keyword"
255       },
256       "query": {
257         "type": "percolator"
258       },
259       "knn": {
260         "type": "flattened"
261       },
262       // page fields
263       "url": {
264         "type": "text"
265       },
266       "title": {
267         "type": "text"
268       },
269       "body_content": {
270         "type": "text"
271       }
272     }
273   }
274 }
275
276
277 GET user-queries/_search
278 {
279   "query": {
280     "percolate": {
281       "field": "query",
282       "index": "vector-search-elasticsearch-fu-finder-pages",
283       "id": "63d3a0d21238d1060f3b9ca3"
284     }
285   }
286 }
```

```
1  {
2   "took": 19,
3   "timed_out": false,
4   "_shards": {
5     "total": 1,
6     "successful": 1,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": {
12      "value": 1,
13      "relation": "eq"
14    },
15    "max_score": 0.23289725,
16    "hits": [
17      {
18        "_index": "user-queries",
19        "_id": "JSFXuYsBmLtfKjaC8afi",
20        "_score": 0.23289725,
21        "_source": {
22          "document_id": "63d3a0d21238d1060f3b9ca3",
23          "query": {
24            "multi_match": {
25              "query": "susan",
26              "fields": [
27                "title",
28                "body_content"
29              ]
30            }
31          },
32          "knn": {
33            "field": "text_embedding.predicted_value",
34            "k": 10,
35            "num_candidates": 50,
36            "query_vector_builder": {
37              "text_embedding": {
38                "model_id": "sentence-transformers__msmarco-minilm-l-12-v3",
39                "model_text": "susan"
40              }
41            }
42          }
43        }
44      }
45    ]
46  }
```

Elasticsearch Client



index-query-elasticsearch.ts

```
const userQueryIndex = 'user-queries';

const cloudID = process.env.ELASTIC_CLOUD_ID;
const apiKey = process.env.ELASTIC_API_KEY;

const client = new Client({
  cloud: { id: cloudID },
  auth: { apiKey: apiKey },
});

export async function indexGameSearch(documentID: string, searchTerms: any) {
  let document = {
    document_id: documentID,
    query: searchTerms.query,
    knn: searchTerms.knn
  };

  return client.index({
    index: userQueryIndex,
    document: document
  });
}
```

Let's Play!

The screenshot shows a smartphone displaying the elastic Fu-Finder app. The top bar is black with the elastic logo and "Fu-Finder". Below it is a pink header with a clock icon showing "0:47" and a piggy bank icon with "0". The main content area shows a search result for "Kate Moss" on IMDb. It includes a thumbnail image of her, a video player showing a clip from "Quant (2021)", and some text about her career. At the bottom, there is a code editor window with the following Elasticsearch query:

```
1 // fields: url, title, body_content
2 {
3     "query": {
4         "match": {
5             "title": "My page query"
6         }
7     }
8 }
```

At the very bottom, it says "Made by Carly Richmond 🎉 with ❤️ and 💪".



SCAN ME

The screenshot shows a desktop browser window with the elastic Fu-Finder app. The top bar has the elastic logo and "React App". The main content area shows a search result for "Kate Moss" on IMDb. It includes a thumbnail image of her, a video player showing a clip from "Quant (2021)", and some text about her career. On the right side, there is a code editor window with the same Elasticsearch query as the smartphone version. At the bottom, there are two sections: "Results" which says "None yet..." and "No results available", and "Made by Carly Richmond 🎉 with ❤️ and 💪".

```
1 // fields: url, title, body_content
2 {
3     "query": {
4         "match": {
5             "title": "My page query"
6         }
7     }
8 }
```

At the bottom, it says "Made by Carly Richmond 🎉 with ❤️ and 💪".

DANKE !

THANK YOU !

MERCI !

GRAZIE !

GRACIAS !

DANK JE WEL !

