Deploying a Cost-Effective and Production-Ready Deep News Recommender System in the Media Crisis Context

Jean-Philippe Corbeil
Le Devoir
Canada
jpcorbeil@ledevoir.com

Florent Daudens
Le Devoir
Canada
fdaudens@ledevoir.com

ABSTRACT

In the actual context of the media crisis, online media companies need cost-effective technological solutions to stay competitive against huge monopolistic software companies massively feeding content to users. News recommender systems are well-suited solutions, even if current commercial solutions are well above most online media's budget. In this paper, we present a case study of our deployed deep news recommender system at *Le Devoir*, an independent french Canadian newspaper in the province of Quebec. We expose the software architecture and the issues we have met with their solutions. Furthermore, we present four qualitative and quantitative analyses done with our custom monitoring dashboard: offline performances of our models, embedding space analysis, fake-user testing and high-traffic simulations. For a tiny fraction of the available commercial solutions' prices, our current simple software architecture based on the Docker, the Kubernetes and open-source technologies in the cloud has demonstrated to be easily maintainable, scalable, and cost-effective. It also shows excellent offline performance and generates high-quality embeddings as well as relevant recommendations.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Recommender System, News Recommendations, Sequential Recommendation, Media Crisis, Dashboard, Cloud Technology

Reference Format:
Jean-Philippe Corbeil and Florent Daudens. 2020. Deploying a Cost-Effective and Production-Ready Deep News Recommender System in the Media Crisis Context. In *3rd Workshop on Online Recommender Systems and User Modeling (ORSUM 2020), in conjunction with the 14th ACM Conference on Recommender Systems, September 25th, 2020, Virtual Event, Brazil.*

1 INTRODUCTION

In the last couple of decades, the newspapers have seen their world changed by the digital shift in the news market [4, 21, 23]. From the newspapers to the online articles, the readers' needs have also shifted from the static paper format to the fast, dynamic and well-synthesized display of the online news on mobiles [1, 13]. Moreover, we have observed the colossal impact — financially and on the behaviours of readers — of intermediary platforms such as news aggregator platforms and social networks that massively feed news content to readers in a personalized fashion.

In this context, the online media companies need digital tools to retain their readers on their platform and augment their conversion goals. We can address all these issues by feeding a personalized list of contents to the readers, making news recommender systems the perfect solution. Nevertheless, most of the commercial data solutions like *Google Analytics 360* and *Google Recommendations AI* are far beyond what most newspapers can afford. Can we build a production-ready and cost-effective deep news recommender system for news articles that leverage the cloud and open-source technologies? At *Le Devoir* — an independent french Canadian journal in the province of Quebec —, the conception and deployment of this recommender system is a part of our digital shift plan. It is also a part of our solution to reach our marketing goals by offering tailored redactional content to our readers.

In this paper, our contributions are:

- The first case study on the deployment of a production-ready cloud architecture of a recommender system with the docker technology and a continuous integration and continuous deployment (CI/CD) production cycle.
- The design of a cost-effective and scalable deep news recommender system oriented on short-term recommendations.
- The demonstration of a considerable offline performance while meeting our constraints: cost, scalability, training time and serving time.
- The design of two qualitative experiments to assess the quality of a news recommender system before going online: the embedding quality test and fake-user recommendation test.
- The design of a monitoring dashboard for our recommender system.

In the next section, we discuss the previous works related to our current news recommender system. Then, we elaborate on our system architecture by explaining all the aspects of our methodology: data processing, model training, recommendation delivering and monitoring. Then, we discuss our model's offline results, two qualitative validation methods and the traffic benchmark of our system. Afterwards, we talk about our system limitations and future works. We end with a conclusion summarizing our whole approach.

2 BACKGROUND

Garcin et al. (2013) [5] presented the Pen recsys, a framework for a news recommender system, made with Java EE. They presented five models in an A/B testing setup: context-tree recommendations, most-popular recommendations, content-based recommendations, collaborative recommendations and random recommendations. They had a web-based control panel to monitor the performances and to modify some parameters. They demonstrated the ability to deliver recommendations within 30 ms even at visit peaks. We followed many aspects of their framework: an A/B testing setup with many models, a monitoring dashboard and a traffic benchmark. Many aspects of their approach are yet dubious, given the recent state-of-the-art in software engineering. First, the current practice in software maintainability is far beyond their Java EE setup. We used cloud-based and open-source technologies with a CI/CD pipeline actionable by our GitHub repositories. Second, their web-based control panel was very minimalist and did not follow any design principles. We built our user interface based on the dashboard principles of Sarikaya et al. (2018) [20], dividing it into six tabs targeting specific monitoring and decision-making goals. We included features like complete monitoring visualizations, actionable widgets and live tests of the models. Third, the sequential nature of the news recommendation is known to require deep sequential architecture [24]. Our collaborative deep learning models are based on Pytorch [19] and take as input sequences. We also prepare an A/B testing setup to test our models against the most-popular recommendations and random ones. As future works, we plan the conception of a content-based approach and its hybridization with our collaborative approach. Finally, we discussed the financial impact of our system in terms of costs for our newspaper company.

Karimi et al. (2018) [10] reviewed state of the art concerning news recommender systems and made several suggestions. They noticed that the scalability of systems is often an issue despite the maturity of storing systems. An issue reported with deep learning architecture by Zhang et al. (2019) [24] as well. Instead of following their recommendation of using continuous learning, we suppressed this issue by designing our system on a few-days basis. With this peculiar design choice, we were able to remove the model dependency on continual indexations of both items and users into a stable model. Second, a major issue concerns the dynamic addition of publications throughout the day and the need to consider these articles quickly. Karimi et al. mentioned the need to incorporate new articles within minutes to benefit from its momentum resulting in a high click-rate-through. We followed their recommendation with many quick training sessions within one hour. They also proposed a hybrid solution mixing both efficient short-term predictions and sophisticated long-term predictions — in line with the previous works of Liu et al. [14]. We adopted this strategy in our system. In the current paper, we are focusing on short-time predictions with our news recommender system. The authors added critics about the reproducibility in the news recommendation domain with many proprietary datasets. In the context of our paper, we addressed this issue by releasing the anonymized collaborative dataset we used in our experiment. Finally, they mentioned the lack of correspondence between offline and online results demonstrated by Garcin et al. (2014) [6]. In our case, we reported the results on our released dataset with further results from training the model many times on evolving data. Despite relying only on offline results, our architecture is very flexible. Even if such correspondence does not happen when we start to provide recommendations online, we have our custom dashboard containing widgets to modify the models, its hyperparameters and the recommendation strategies in an A/B testing setup. Furthermore, we design our system with a CI/CD pipeline enabling us to make more profound modifications ready to be released within the next ten minutes without any downtime.

Mohallick and Özgöbek (2017) [16] analyzed privacy in news recommender systems. Despite many reported news recommender systems relying on personal data, our system is in line with privacy principles. We do not use more data than the required information for a collaborative algorithm. We only need to activate the opt-out option of *Matomo* on our website to be fully GDPR compliant.

3 METHODOLOGY

3.1 Architecture

We divided the dynamic of our system into two segments the analytics part and the recommender system part. We illustrated these two parts in Figure 1.

The analytics side contains two technical components: the analytics platform *Matomo* [15] (previously *Piwik*) and the web site, in our case *ledevoir.com.* As a first step, the readers come to our website, and we track their page views anonymously. We respect our readers' privacy by assigning a random visitor id at the first visit and by having no specific personal information[1]. Only a truncated IP at 2 bytes — which is very general information — is recorded, and no more than three months of detailed data is available locally. At *Le Devoir*, the marketing department manages any other information linked to the user accounts on a different CRM system. As a second step, we added the recommender system part in the loop to propose a personalized list of five articles.

Overall, we designed our recommender system with five essential components: a serving virtual machine, a training virtual machine, a GCP bucket, a MongoDB database and a monitoring dashboard. We explained the interactions between all the components in the following sections. We also address specific design considerations.

3.2 Design considerations

Our recommender system's design must respect four significant constraints: the cost, the scalability, the serving time and the training time. Thus, we made two major design choices: split our architecture into two virtual machines and limit the number of days.

We chose to split our architecture into two virtual machines (VM): one Serving VM and one Training VM. The first VM is the master VM. It is always online, and it serves recommendations to our readers. The second VM has the only job of training the model. We made this peculiar design choice to reduce the cost since the *Graphics Processing Unit* (GPU), needed to train the model in a reasonable amount of time, is the most expensive part of the system. With this peculiar design choice, we estimated based on the *GCP Pricing Calculator* to save near 80% of our total cost considering 5.5 hours of training per day. This training time is wisely split into training sessions of about 7 to 10 minutes, with, on average, 4.8 minutes only to train the model (taken on our dashboard's *current state* and *model* tabs between May 29th, 2020 and May 31th 2020).

[1]We followed *Matomo*'s guideline on privacy: https://matomo.org/docs/privacy/.

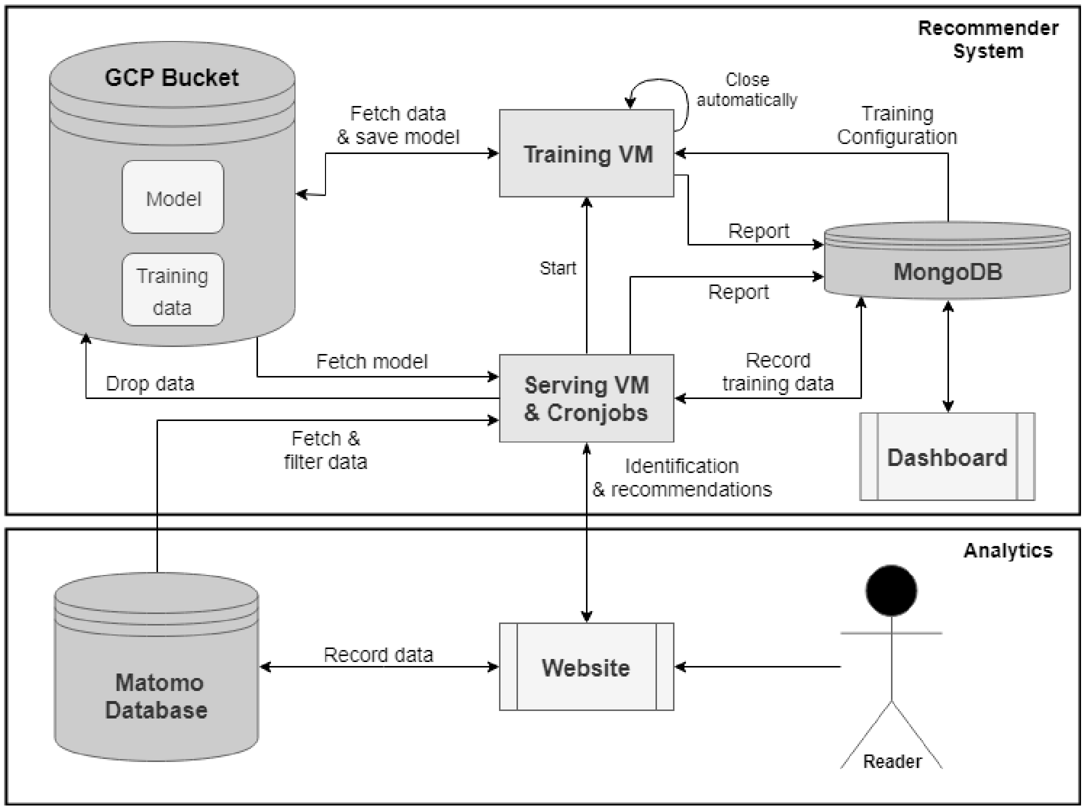


Figure 1: Diagram of our recommender system architecture.

Thus, we can have a maximum of 6 training sessions by hours of the day, leading to model updates of the same frequency on the serving VM. We designed the number of training sessions to follow the averaged daily traffic on our website in Figure 2 with some delays. We apply this strategy to follow the principle of training more often when more new data are available.

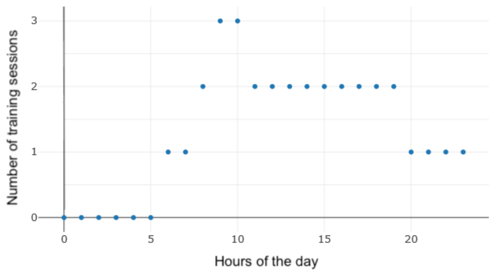


Figure 2: Interactive plot taken from our dashboard with daily number of training sessions by hour of the day.

Second, we selected a maximum number of days to train and from which we recommend articles. This design choice would make the amount of input data steady and stabilize the system training time. Furthermore, we reduced the overall dependency of the system on keeping the previously trained model and maintaining the indexations leading us to a scalable architecture with a simpler sequential model that can be trained every time from scratch. We fixed this number at four days, knowing that our articles' active lifespan is at most two days following the Nyquist rate [2, 17]. On average, we then have around 1.9 million data points for each training session (taken on our dashboard's *model* tab from May 29th, 2020 to May 31st, 2020).

3.3 Data analytics

To develop a news recommender system, we needed the right analytics platform to have the necessary insight into our data. Two major drawbacks from the widely used *Google Analytics 360* is its expensiveness and its data sampling. Thus, we solved both issues by implementing the open-source *Matomo Analytics Platform* [15] with a MySQL database. It is freely available, has a great community and is easily deployable on any cloud computing virtual machine. Thus, we implemented *Matomo Analytics Platform* on *Google Cloud Platform* (GCP) for our high-traffic website.

3.4 Data pipeline

From Matomo's database, we designed an intermediary MongoDB database on *MongoDB Atlas*. This database is a crucial piece of our design because our website's traffic heavily solicits our central database. This secondary database holds around a week of data already pre-processed, and it is fed at every minute by cronjobs on the serving VM. Because Matomo records noisy data such as 404 URLs and pages that are not articles on our website, we had to filter our data. We conceive it to consider the data with the right URL format for articles and to validate its title before dumping the results into our MongoDB. We also separated the article records (URL, title, and ID) from the visit records (ID, article ID, reader ID, and timestamp) to maintain a lower memory usage. Even with this architecture, the download speed of the whole data at training time was slow on the training machine — around 10 minutes. This downtime can be costly on a virtual machine with a GPU, such as our training virtual machine. Our solution was to pre-dump the data progressively into a Google Cloud Bucket and download this file instead, which resulted in a download time of less than a minute. We kept the MongoDB in this part of our design for real-time access to accelerate our cronjobs pre-processing and to maintain our data integrity.

3.5 Models

At the core of our system, we have implemented the Spotlight python library by Kula et al. [12]. It contains many state-of-the-art deep sequential recommender system architectures in PyTorch [19]. We used their sequential models, which includes four neural models: 1D CNN [9, 18], LSTM [7], MixtureLSTM [11] and Pool model [3].

We optimized our models with actual offline data from May 7th, 2020, to May 11th, 2020, containing 1,944,719 data points. We made an anonymous version of this dataset on our GitHub[2] to promote our results. We encourage the community to improve our results with better collaborative models. The data contains records with anonymized reader identification number, anonymized article identification number, and timestamp.

In our experimentation, we fixed some parameter values according to both our pre-experiment and Spotlight's documentation: the number of epochs to 10, the learning rate to 1e-2, the random state to 42, and we used no regularization. We took the adaptive hinge loss function [22]. We did our experiments with an NVIDIA RTX2070 GPU.

With Spotlight's sequence parser, we parse all sequences of articles with a minimum of 3 articles and a maximum of 7 articles for each reader. Every sequence is also padded up to 7 articles. We chose these bounds first to ensure that the sequences contain a minimum of relevant articles and second to limit the length of the model's input. For the lower bound of three articles, we have a significant part of our traffic that only consults one or two article(s) and does not come back. We do not aim at recommending articles to this type of reader. We prefer to recommend to our core readers at first. For the upper bound, we did pre-experimentation, and seven articles seemed a reasonable length. The Spotlight's documentation suggested five items as an upper bound, but it is short given our lower bound.

Since we have a large number of sequences, we did our validation with a training/testing split of 90%/10%, which resulted in a train set of 215,200 samples and a test set of 22,625 samples. We separated these sets according to users. Thus, readers in the train set and test set are mutually exclusive. We optimized all four models with the following set of hyperparameters applying a grid search approach:

- Batch size = { 512, 1024, 2048, 4096, 8192 }
- Embedding size = { 32, 64 }
- Number of negative samples = { 100, 200, 300 }

3.6 Model dumping

Once the model trained on the training virtual machine, we dump its weights and configuration into our *Google Cloud Bucket*. Then, the training machine notifies by HTTPS protocol the serving machine to fetch the new model and to make it ready to serve. Finally, the training machine can turn off until the serving machine calls the next training session.

3.7 Continuous Integration and Continuous Deployment Pipeline

We followed the *Continuous Integration and Continuous Deployment* cycle principle from the software engineering field to ease the maintainability of our system and to ensure zero downtime. Our pipeline is illustrated in Figure 3. We hosted our recommender system in a private GitHub repository. We activated a trigger to watch for updates on the master branch and to call our continuous integration (CI) platform *CircleCI* automatically on this event. The CI runs the recommender system's unit tests, which cover our code at 100% to ensure strict control of our builds. If the tests are all passed, the CI build from our repository a Docker image and registers it on *Google Container Registery* (GCR) afterwards. Finally, the CI deploys the new image into our Kubernetes cluster[3] with zero downtime by applying a rolling update. The serving VM and the training VM are processed similarly by two different CI/CD pipelines linked to their respective GitHub's master branch. The only difference is that the training image runs as a Kubernetes Job on its cluster since we run it on demand.

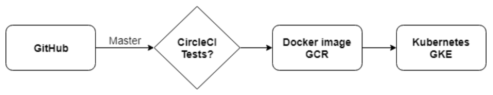


Figure 3: Diagram of the CI/CD pipeline of the recommender system.

3.8 Online validation strategy

To validate the model effectiveness when we launch it online, we prepared an A/B testing setup inside the recommender system by monitoring the Click-Through-Rate (CTR) distributions. In this A/B testing, we compare our model's recommendations to the actual top-5 article suggestions given to the readers in a box on the website. Our model's recommendations are given in a similar box with the same disposition — see Figure 4. We decided that half the readers get the model's recommendations, and the other half get recommendations using the best articles in the last 30 minutes. By applying the Student's T-test, we can measure the relevance of one

[2]https://github.com/LeDevoir/orsum2020_collaborative_datasets

[3]Kubernetes is a system for automating deployment, scaling, and managing Docker containers.

recommendation method over the other. Moreover, we can change the A/B testing settings with our custom dashboard.

3.9 System monitoring

Many metrics and variables are recorded on the MongoDB to monitor and control the system once online. This monitoring is done with a custom dashboard made in *Dash* by *Plotly* [8]. Our prototype has six different tabs: current state, performances, model, execution settings, embedding space viewer and model testing interface. In Figure 5, we have included an illustration of the *current state* tab. We use it to monitor the training VM status, which is useful to detect errors during training. We also use it to see the current A/B testing results with two histograms with its T-test values. The *performances* tab is designed to monitor the current model offline performances (MRR and P@5, see section 4.1). The *model* tab displays the training time and the number of input data across time for monitoring purposes. The *execution* tab has many variables to manipulate and modify the system. We can shut down the recommendations, change the A/B testing settings, change the number of recommendations and interact with a plot to set the number of training sessions per hour of the day. The *embeddings* tab (see Figure 7) is a live 3D plot of the embeddings space generated by the trained model compressed in 3 dimensions with the TSNE algorithm. The *test* tab (see Figure 8) contains an interface to test the recommendations of the current model. It has the list of all current articles from which to make a selection. Then, this selected list is sent to the server as a list of previously consulted articles by a fake reader. We send back the recommendations of the model. The last two tabs are interesting tools to evaluate the model recommendations (see Sections 4.2 and 4.3).

4 RESULTS

4.1 Offline Model validation

Out of 120 possible experiments in our grid search, 12 finished with an "out of GPU memory" error leaving us with 108 results. We kept the top-5 for each model in Table 1. By fine-tuning the models' hyperparameters, we found that the 1D CNN architecture is the most optimal one for our task, followed by the LSTM. Since it is faster to train than the LSTM by 38 seconds and saves memory with smaller embeddings, we selected the CNN model as our first state-of-the-art configuration. Overall, the MixtureLSTM model takes more time to train for about slightly lower results than the LSTM and the CNN. The Pooling model is largely under-performing based on MRR, but competitive with the P@5 metric. We found that a large negative sampling is improving the results as well as a large embedding size in general. Moreover, smaller batch sizes tend to get better results. For instance, we reported no batch size of 8192 in these top-5 and only one 4096. Most of the 20 results presented here obtain similar P@5, which means that in a list of 5 articles, we found most of the time one relevant article for our reader. The MRR score indicates that, except for the Pooling model, most models suggest this relevant article as first or as the second article, between 1 and 0.5 respectively.

We supported the generalization of these results by providing the histogram of MRRs measured from 22 training sessions between May 26th, 2020 and May 28th, 2020, in Figure 6. We took this Figure from our live dashboard's *performances* tab. The input data given to our 1D CNN model is live evolving data from a window of the previous four days. We see a steady MRR performance across the time of 0.69 ± 0.03 in line with current results when considering an error based on three standard deviations.

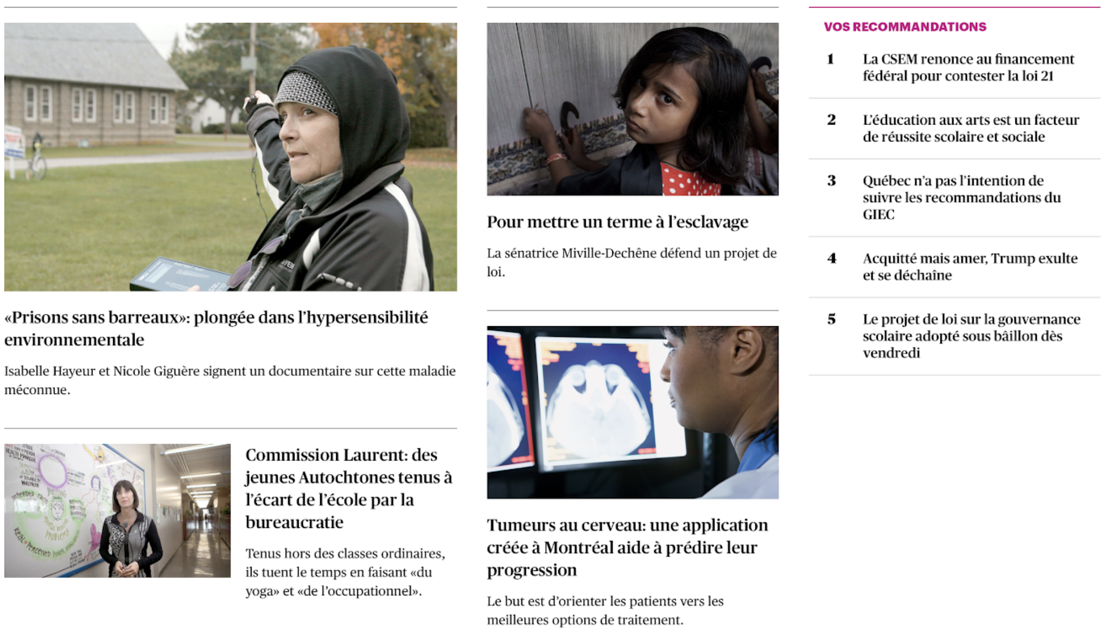
4.2 Analysis of article embedding's quality

In Figure 7, we analyzed the embedding space of our 1D CNN with an embedding size of 32 dimensions using the TSNE algorithm to project them into a 3D space. In Figure 7a, we took the data of May 20th, 2020, for five subjects: world, politics, culture, opinion and economy. We did not consider the society subject, for which the scope is vast and too similar to many other subjects. It would have made the Figure harder to visualize. We omitted the lecture and lifestyle subjects too because they usually contain only a couple of articles each. By illustrating the subjects with colours for each article, we notice clusters linked to the article's subject. The existence of these clusters indicates that the model has learned relevant representations for the articles. For instance, we distinguish the opinion cluster (blue) and the culture cluster (green) on the left and right of the Figure, respectively. We argue that this is due to their different nature in the writing style and in the subject, which attracts different readers. We also see the politic cluster (red) in the center near the world cluster (gray) and the economy cluster (orange). We argue that these three subjects are mostly related and have a similar writing style, which attracts similar readers. Since we integrated this view on our dashboard, we can further confirm the appearance of similar cluster patterns emerge mostly every day — see Figure 7b.

We know that the model learns the embeddings from our collaborative data. Thus, they also integrate the influence of their locations on our website partly. Thus, we plan to use this embedding viewer and its dynamics as a management tool for our website display as future work.

4.3 Analysis of fake-reader recommendations

We further tested and analyzed our recommender system's recommendations with the user interface in Figure 8 from our custom dashboard. On this interface, we have a tool to send our recommender system model a list of selected articles as an input and to receive its recommendations. We design the experiment to assess the quality of our recommendations. We selected two articles from each five previously selected subjects: world, politics, culture, opinion and economy. Then, we submitted two articles to the system and analyzed the five recommended articles. The results are in Table 2. First, our results show that our model tends to recommend articles from the same subject about half the time in our samples (13 out of 25 articles). We see strong links between the most recommended articles and submitted articles. For instance, in the world sample, we see both articles are about the COVID-19 pandemic and that the last article is related to Trump. In the recommended articles, we received a complete list of COVID-19 and the fourth recommendation about Trump as well. Moreover, in the culture sample, we see that the last submitted article is about a TV show named "Occupation Double" and the first recommendation is about the same TV show "OD" as well as the fourth recommendation.



«Prisons sans barreaux»: plongée dans l'hypersensibilité environnementale

Isabelle Hayeur et Nicole Giguère signent un documentaire sur cette maladie méconnue.

Commission Laurent: des jeunes Autochtones tenus à l'écart de l'école par la bureaucratie

Tenus hors des classes ordinaires, ils tuent le temps en faisant «du yoga» et «de l'occupationnel».

Pour mettre un terme à l'esclavage

La sénatrice Miville-Dechêne défend un projet de loi.

Tumeurs au cerveau: une application créée à Montréal aide à prédire leur progression

Le but est d'orienter les patients vers les meilleures options de traitement.

VOS RECOMMANDATIONS

1 La CSEM renonce au financement fédéral pour contester la loi 21

2 L'éducation aux arts est un facteur de réussite scolaire et sociale

3 Québec n'a pas l'intention de suivre les recommandations du GIEC

4 Acquitté mais amer, Trump exulte et se déchaîne

5 Le projet de loi sur la gouvernance scolaire adopté sous bâillon dès vendredi

Figure 4: Implementation on our website of the recommendation box "VOS RECOMMANDATIONS" on the right.

Table 1: Top-5 results for each model sorted by MRR.

Model	Embedding size	Batch size	Negative samples	MRR	P@5	Training time (s)
CNN	32	2048	300	**0.693**	**0.133**	**134**
LSTM	64	1024	300	0.693	0.133	172
LSTM	64	2048	300	0.693	0.133	157
LSTM	64	512	300	0.691	0.133	219
CNN	64	2048	300	0.690	0.132	156
CNN	32	1024	300	0.690	0.132	154
MixtureLSTM	32	512	300	0.689	0.131	549
LSTM	64	512	200	0.687	0.131	159
LSTM	64	2048	200	0.687	0.123	108
CNN	64	2048	200	0.687	0.131	107
MixtureLSTM	64	512	200	0.687	0.122	614
CNN	64	4096	100	0.684	0.132	54
MixtureLSTM	32	1024	200	0.682	0.130	350
MixtureLSTM	32	512	200	0.680	0.128	381
MixtureLSTM	64	512	100	0.677	0.125	334
Pooling	64	512	300	0.478	0.131	211
Pooling	64	1024	300	0.478	0.133	167
Pooling	64	2048	300	0.477	0.133	155
Pooling	64	2048	200	0.476	0.132	106
Pooling	64	1024	100	0.476	0.132	65

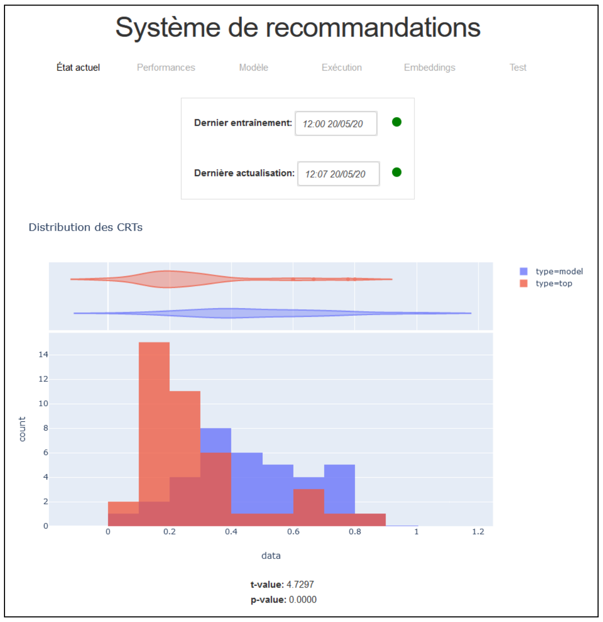
Second, it is also interesting that not all articles are from the same subject, which gives some serendipity to our recommendations. From our observations with our website, we noted that the model tends to use a good recommendation strategy by recommending a list of articles mixing top-viewed ones and subject-related ones.

4.4 Traffic benchmark

We developed a script that repeatedly sends to the serving VM real user identification numbers to benchmark the maximum traffic

Table 2: Five fake user tests done on May 20th, 2020. We send two submitted articles as a fake list of previously read articles to our recommender system. Then, we receive five recommended articles. The names of the articles are in french.

Subject	Submitted articles	Recommended articles	Section	Result
Economy	1. L'intelligence artificielle au service du transport de conteneurs	1. La COVID-19 fera fléchir les prix immobiliers	**Economy**	
	2. Forte baisse des ventes de maisons au Québec	2. Situation préoccupante pour les ménages très endettés	**Economy**	
		3. Une deuxième vague «inévitable» au Canada, disent les experts	Society	**3/5**
		4. Masque sanitaire et burqa: une insulte à l'intelligence	Opinion	
		5. Refroidissement immobilier	**Economy**	
Culture	1. Décès de la comédienne Michelle Rosignol	1. Après OD, la vie	**Culture**	
	2. Occupation double: rattrapé par la réalité des beaux sentiments	2. Emmener Google au théâtre, et vice versa	**Culture**	
		3. F1: Lawrence Stroll met la barre haute pour Aston Martin dès 2021	World	**3/5**
		4. Aimer résister à Occupation double	**Culture**	
		5. Les mots de l'année (6/6): «Fake news», les vraies fausses nouvelles	Society	
World	1. Quel est le bilan véritable de la pandémie de COVID-19?	1. Coronavirus: un «mini-Schengen» se prépare en Europe pendant que d'autres pays se referment	**World**	
	2. Un président qui défie la science	2. Le masque non médical protège-t-il celui qui le porte?	Society	
		3. Interdit ou pas avec le déconfinement?	Society	**2/5**
		4. L'«incompétence» de Pékin a provoqué une «tuerie de masse mondiale», selon Trump	**World**	
		5. Les libéraux n'ont pas respecté leurs promesses, accuse Blanchet	Politics	
Opinion	1. L'éclatant succès de Taïwan	1. Quel est le bilan véritable de la pandémie de COVID-19?	World	
	2. Élèves abandonnés, parents épuisés	2. Le masque non médical protège-t-il celui qui le porte?	Society	
		3. D'égal à égal, le Québec, 40 ans plus tard?	**Opinion**	**2/5**
		4. Tout est affaire de décor pendant le confinement	Society	
		5. Référendum 1980 – l'étrange campagne de sécurisation	**Opinion**	
Politics	1. Une nouvelle aide fédéral pour les PME	1. Interdit ou pas avec le déconfinement?	Society	
	2. Le Québec déplore 51 nouveaux décès dus à la COVID-19	2. Pincez-moi, Docteur Horacio, je rêve…	Opinion	
		3. Feu vert pour la réouverture des commerces à Montréal	**Politics**	**3/5**
		4. Trois artères de Rosemont-La Petite Patrie fermées aux voitures	**Politics**	
		5. La frontière entre le Canada et les États-Unis reste fermée jusqu'au 21 juin	**Politics**	



Dernier entraînement: 12:00 20/05/20 ●

Dernière actualisation: 12:07 20/05/20 ●

t-value: 4.7297
p-value: 0.0000

Figure 5: Example of our custom dashboard's user interface in french. It is the *current state* tab ("état actuel") on our recommender system. We see our six navigation tabs: current state, performances, model, execution settings, embedding space viewer and model testing interface. In the first box below our navigation, we have two indicators about the training status (last training status and model reloading status). In the bottom, we have two CTR histograms that help us visualize the current A/B testing result with their T-test values below — these are the results of our traffic benchmark (see Section 4.4).

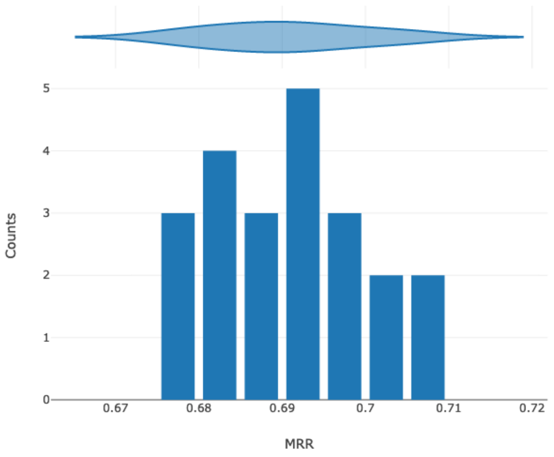
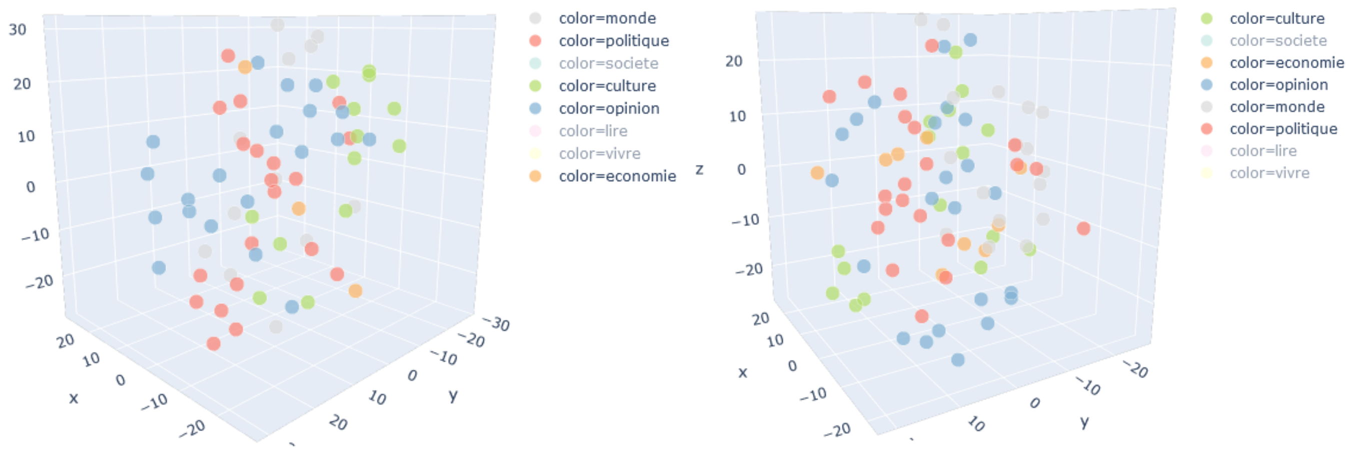


Figure 6: Distribution of MRR for our 1D CNN model, measured from May 26th, 2020, to May 28th, 2020, across 22 training sessions. We took this Figure from our dashboard (*performances* tab) on May 28th, 2020.

supported by our current configuration. With each received recommendation list, we made a rule-based decision process to simulate the click rate. If the recommendations come from the model (*model*), we click on any recommendation with a probability of 1/2. If we recommended using the top-viewed articles in the last 30 minutes (*top*), we would click on any recommendation with a probability of 1/4. We fixed the number of readers per second at 3 with a pool of 2000 readers. The fake A/B testing results are on our dashboard's current state tab in Figure 5 — with very high significative results. The CTR distributions are very close to our rule-based decision process. We also measure the time elapsed between sending the request and receiving the response, which we displayed in the histogram of Figure 9. We see that our recommendations are made in three seconds on average, which is okay since we feed our recommendations box asynchronously. Knowing that our articles' average reading time is close to one minute, we have the time to fill the box with recommendations since it appears half-way. We also looked at the correlation between the order in which we sent the requests and the time lapses, and we can report correlation of less than 0.05. Since our morning peak hour has about 2.5 readers per second, our current configuration is ready to feed our website in real-time. Compared to Garcin et al. [5], which had a response of 30 ms with a Java EE architecture, we note that our system is slow. We argue that this is due to our serving VM in Python, serving with Flask through gunicorn — known to be slower than Java. We also have deep learning models served on CPU — which is a specific design choice. Since we are meeting all constraints, we leave the service response time optimizations to future works.

5 LIMITATIONS AND FUTURE WORKS

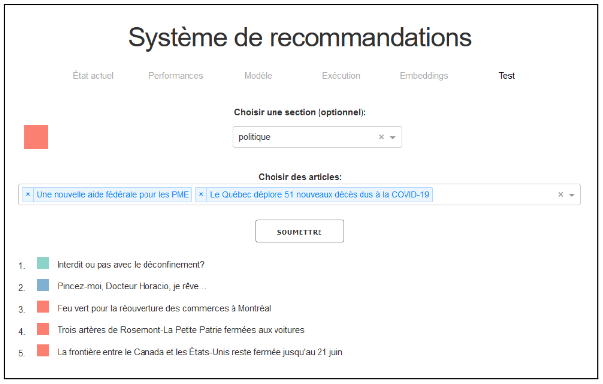
In our case study, many aspects have limitations and need further improvements. We have short-term recommendations for the limitations of the recommendations, small grid-search optimization, the offline performance metrics (MRR and P@5), and the cold-start issue. We chose to work on short-term recommendations to improve the scalability of our system. We will make further developments in the future to include long-term recommendations by adding a content-based approach. We limited our approach to a small grid search to optimize our models' hyperparameters. We chose a small set of values for each hyperparameter to train the models in a reasonable amount of time based on insights from our pre-experiments. We obtained good results, and we hope that other researchers will try their approaches on our released dataset to push our state-of-the-art. While the combination of both MRR and P@5 is relevant, the first is limited to measure the appearance of the first relevant item in the list, while the second is the proportion of relevant items in the whole list. Since we chose these metrics because of our current lack of specific relevance measurements, we plan on extracting the reading time of articles from *Matomo* to compute the relevance of our article for a given reader as recommendations. Then, we will compute the NDCG@5, which is a better metric to evaluate our offline performance. Finally, we also face the cold-start issue that we did not address directly. Nevertheless, we design the recommendation box to appear only in the article because of our short-term recommendations. Therefore, readers coming for the first time on our website will still get recommendations when they visit articles.



(a) May 20th 2020.

(b) May 26th 2020.

Figure 7: Embedding projection of our article embeddings taken from our custom dashboard (*embeddings* tab) projected in 3 dimensions with TSNE and colored for 5 main sections of our website: world, politics, culture, opinion and economy.



Système de recommandations

Choisir une section (optionnel):

🟥 politique

Choisir des articles:

Une nouvelle aide fédérale pour les PME; Le Québec déplore 51 nouveaux décès dus à la COVID-19

1. 🟩 Interdit ou pas avec le déconfinement?
2. 🟦 Pincez-moi, Docteur Horacio, je rêve...
3. 🟥 Feu vert pour la réouverture des commerces à Montréal
4. 🟥 Trois artères de Rosemont-La Pette Patrie fermées aux voitures
5. 🟥 La frontière entre le Canada et les États-Unis reste fermée jusqu'au 21 juin

Figure 8: Example of our fake-user test on our custom dashboard — done on May 20th, 2020. We use the first dropdown list to filter articles by subject, which has a pre-assigned colour (e.g. red for politics). With the second dropdown, we can select the articles consulted by the fake user. The result is a list of five recommended articles displayed with subject colours at the bottom.

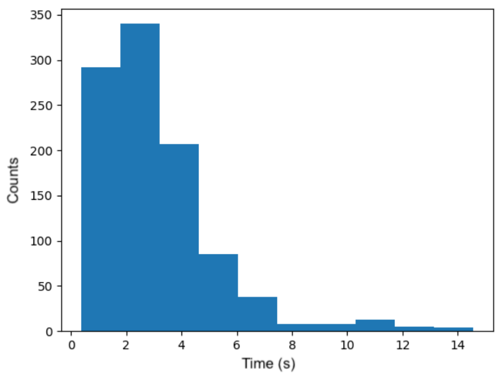


Figure 9: Histogram of time lapses before receiving recommendations. This traffic simulation is done with 2000 readers and a rate of 3 readers per second.

We also have limitations with our embedding space study and fake-user test study. Their main limitation is their generalization. However, we argue that both studies are complementary and insightful in their results, indicating the learning of both relevant embeddings and relevant recommendations. We also demonstrated the same observations in the embedding space for both May 20th, 2020, and May 26th, 2020.

6 CONCLUSION

To conclude, we presented a case study about our cost-effective and production-ready deep news recommender system architecture with open-source and cloud technologies. We designed it — with two virtual machines (serving VM and training VM) and with a limitation on the number of days — to meet our cost, scalability, training time and serving time constraints. With a grid-search approach, we found that the optimal model was the 1D CNN performing with an MRR of 0.693 and a P@5 of 0.133 in only 134 seconds. We release an anonymized version of our dataset to promote the reproducibility of our results. In our architecture, the model is trained from scratch in many training sessions wisely distributed according to our website traffic. We estimated this strategy to save around 80% of our total cost for the recommender system, which is less than 4$US per day. Compared to commercial solutions costing thousands of dollars per month, this percentage rises close to 98 %. We also evaluated our systems with two more studies. Using our custom monitoring dashboard, we observed a high relevance of our embeddings and recommendations based on two complementary qualitative studies: the embedding space study and the fake-user test study. Finally, we demonstrated the readiness of our system with a traffic simulation. We hope our affordable and robust design inspires other online media companies to consider developing their recommender systems to be competitive in the digital news market.

REFERENCES

[1] Kevin G Barnhurst. 2011. The new "media affect" and the crisis of representation for political communication. *The International Journal of Press/Politics* 16, 4 (2011), 573–593.

[2] Harold S Black. 1953. *Modulation theory.* van Nostrand.

[3] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems.* 191–198.

[4] Marc Edge. 2014. Newspapers' Annual Reports Show Chains Profitable. *Newspaper Research Journal* 35, 4 (2014).

[5] Florent Garcin and Boi Faltings. 2013. Pen recsys: A personalized news recommender systems framework. In *Proceedings of the 2013 International News Recommender Systems Workshop and Challenge.* 3–9.

[6] Florent Garcin, Boi Faltings, Olivier Donatsch, Ayar Alazzawi, Christophe Bruttin, and Amr Huber. 2014. Offline and online evaluation of news recommender systems at swissinfo.ch. In *Proceedings of the 8th ACM Conference on Recommender systems.* 169–176.

[7] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[8] Plotly Technologies Inc. 2015. *Collaborative data science.* Montreal, QC. https://plot.ly

[9] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099* (2016).

[10] Mozhgan Karimi, Dietmar Jannach, and Michael Jugovac. 2018. News recommender systems–Survey and roads ahead. *Information Processing & Management* 54, 6 (2018), 1203–1227.

[11] Maciej Kula. 2017. Mixture-of-tastes models for representing users with diverse interests. *arXiv preprint arXiv:1711.08379* (2017).

[12] Maciej Kula. 2017. Spotlight. https://github.com/maciejkula/spotlight.

[13] Azi Lev-On. 2012. Communication, community, crisis: Mapping uses and gratifications in the contemporary media environment. *New Media & Society* 14, 1 (2012), 98–116.

[14] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. 2010. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces.* 31–40.

[15] Stephan A Miller. 2012. *Piwik web analytics essentials.* Packt Publishing Ltd.

[16] Itishree Mohallick and Özlem Özgöbek. 2017. Exploring privacy concerns in news recommender systems. In *Proceedings of the International Conference on Web Intelligence.* 1054–1061.

[17] Harry Nyquist. 1928. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers* 47, 2 (1928), 617–644.

[18] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499* (2016).

[19] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. *NIPS 2017 Workshop Autodiff* (2017).

[20] Alper Sarikaya, Michael Correll, Lyn Bartram, Melanie Tory, and Danyel Fisher. 2018. What do we talk about when we talk about dashboards? *IEEE transactions on visualization and computer graphics* 25, 1 (2018), 682–692.

[21] Paul Starr. 2012. An unexpected crisis: The news media in postindustrial democracies. *The International Journal of Press/Politics* 17, 2 (2012), 234–242.

[22] Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *Twenty-Second International Joint Conference on Artificial Intelligence.*

[23] Dwayne Winseck. 2010. Financialization and the "crisis of the media": The rise and fall of (some) media conglomerates in Canada. *Canadian Journal of Communication* 35, 3 (2010).

[24] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.