

“An Image is Worth a Thousand Features”: Scalable Product Representations for In-Session Type-Ahead Personalization

Bingqing Yu*
Coveo
Montreal, Canada
cyu2@coveo.com

Ciro Greco*
Coveo Labs
New York, NY
cgreco@coveo.com

Jacopo Tagliabue*†
Coveo Labs
New York, NY
jtagliabue@coveo.com

Federico Bianchi*
Bocconi University
Milano, Italy
f.bianchi@unibocconi.it

ABSTRACT

We address the problem of personalizing query completion in a digital commerce setting, in which the bounce rate is typically high and recurring users are rare. We focus on in-session personalization and improve a standard noisy channel model by injecting dense vectors computed from product images at query time. We argue that image-based personalization displays several advantages over alternative proposals (from data availability to business scalability), and provide quantitative evidence and qualitative support on the effectiveness of the proposed methods. Finally, we show how a shared vector space between similar shops can be used to improve the experience of users browsing *across* sites, opening up the possibility of applying zero-shot unsupervised personalization to increase conversions. This will prove to be particularly relevant to retail groups that manage multiple brands and/or websites and to multi-tenant SaaS providers that serve multiple clients in the same space.

CCS CONCEPTS

• **Information systems** → **Query suggestion**; *Language models*; *Personalization*.

KEYWORDS

e-commerce query auto-completion, conditional language model, neural networks, zero-shot learning, transfer learning

ACM Reference Format:

Bingqing Yu, Jacopo Tagliabue, Ciro Greco, and Federico Bianchi. 2020. “An Image is Worth a Thousand Features”: Scalable Product Representations

* Authors are ordered by direct contribution - idea and execution - in the research project.

† Corresponding author.

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. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](https://www.acm.org/permissions).

WWW '20 Companion, April 20–24, 2020, Taipei, Taiwan

© 2020 Association for Computing Machinery.

ACM ISBN 978-1-4503-7024-0/20/04.

<https://doi.org/10.1145/3366424.3386198>

for In-Session Type-Ahead Personalization. In *Companion Proceedings of the Web Conference 2020 (WWW '20 Companion)*, April 20–24, 2020, Taipei, Taiwan. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3366424.3386198>

1 INTRODUCTION

Type-ahead systems have a long distinguished history in Information Retrieval [4] and are a crucial component of all contemporary state-of-the-art commerce platforms: from the point of view of the retailer, suggesting queries to shoppers when they type in the search bar has the benefit of nudging them into querying for *what* the shop wants them to buy *in the way* the shop wants.

The recent wave of adoption of neural architectures in commerce ([7], [22]) has changed type-ahead systems as well ([12]); *this* work presents a simple-to-implement, principled and effective way to inject real-time personalization into query suggestions in two distinct commerce scenarios: first, we tackle the *within shop* personalization challenge, showing how dense features calculated from product images allow us to quickly compute session vectors that successfully capture shopping intentions; second, we tackle the *cross-shop* personalization challenge, showing how the same deep learning architecture can leverage a shared vector space of product images to transfer the intent from website X to type-ahead on website Y , even when X and Y only have partially overlapping items (as a more extreme case, even when X and Y are in completely different languages).

The *cross-shop* scenario is a typical case of “transfer learning”, and it is a particularly important one for multi-brand groups and, generally, A.I. providers scaling predictions across their network as representing user preferences in type-ahead systems is a formidable challenge for typical mid-size digital shops. The combination of high bounce-rate, low ratio of recurring users, small volume of search queries and inconsistent meta-data make in-session personalization both hard to achieve and *extremely valuable*, as it promises to improve the user experience of a large pool of shoppers and turn them into buyers. Using industry data, we argue that *within shop* and *cross-shop* personalization is a crucial problem to solve in today’s competitive market of digital retailers, and provide substantial evidence that the proposed methods are a substantial improvement over industry best practices. We summarize the main contributions of *this* paper in two major points:

- we extend the noisy channel model for type-ahead, by "injecting" personalization into the language model using dense vectors from product images; we present methods that are easily applicable even to websites with no historical fine-grained user tracking, or languages for which no linguistic resources exist;
- we tackle the problem of scaling personalization *across websites*, as for example across two brands in the same group, or two versions of the same site with different localization: in particular, we show how shoppers' interest can be successfully transferred between websites **X** and **Y** (with no prior data point for the user on **Y**) by leveraging the shared vector space.

This work is structured as follows: Section 2 details our two main motivating use cases from the e-commerce industry; Section 3 discusses relevant previous work on type-ahead, with special focus on neural models; Section 4 gives a high-level description of the target dataset; Section 5 first introduces two ways in which we can evolve the noisy channel model to deal with contextual elements and then explains how to calculate a context vector. Finally, we benchmark the models in Section 6 and provide quantitative and qualitative assessment of the quality of the predictions, before concluding in Session 7 with our product roadmap to engineer these models at scale.

2 TYPE-AHEAD IN PRODUCT SEARCH: A MOTIVATING USE CASE

Consider the browsing patterns of **Shoppers A** and **B** in Figure 1, visiting a digital shop selling sport apparel: **Shopper A**'s latent intent is about "soccer", while **Shopper B**'s is about "tennis". When they type the character "n" in the search bar, the type-ahead service needs to provide them with relevant suggestions, "soccer"-based for **A** and "tennis"-based for **B**. In other words, the ideal language model behind the scene is a *conditional* language model: $P("nadal"|"n")$ is $P("nadal"|"n", I)$, where I is the latent intent of the current user. To strengthen this point, we report in Table 1 some examples of real query from our dataset (Section 4): the amount of overlap in the top queries across website sections is negligible, making "global" probability estimates fairly inaccurate for many shoppers.

Personalization within a session is certainly important, but there is another, often neglected type of personalization - what we call "zero-shot personalization". Consider now **Shopper C** in Figure 2: **C** starts browsing soccer products in **Site 1** and then move to **Site 2** - a different e-commerce site selling sport apparel. **C** has never been on **Site 2**, but we would still like to power her experience with something more than a generic language model for query suggestions: however, we do not have data points for her on **Site 2** and the (unconditioned) language model in **2** may differ significantly from the one on **Site 1**. In an industry where many traditional companies struggle to keep up with the personalization strategies of the top tech players in the space, finding ways to reduce the bounce rate [15] by transferring the learning across different sites can help greatly level the playing fields. Section 5 explains how to transfer context between **Site 1** and **Site 2** to condition the language model on **Site 2** for users which are at the very first interactions with the site itself.



Figure 1: Shopper A and shopper B browse a sport apparel website and then start typing in the search bar: their shopping intent should somehow influence the ranking of possible query completions.

Table 1: Most frequent queries by website section.

Rank	Ski	Women Sneakers	Man Apparel
1	ski trousers	saucony women	octopus sweater
2	man sweater	nike air force 1	sweater
3	man ski trousers	air max 97	octopus
4	ski gloves	nike air max	man sweater
5	ski jacket	running shoes	man overall

Coveo is a SaaS search provider with more than 500 clients. As an industry case (see also our remarks in Section 7), one of our clients has more than a dozen brands selling the same type of goods (i.e. shoes) and, at the same time, very little fine-grained historical interaction data are available. While planning the progressive roll-out of all brands, it became clear that deploying effective machine learning solutions at day one would have been unfeasible, highlighting the business value of applying scalable transfer learning strategies between the websites.

Our main research questions are therefore the following two:

- (1) is there a fast-to-compute notion of "context" for e-commerce type-ahead that can be used to transform a "static" language model into a contextual one?
- (2) can we map "context" from **Site 1** to **Site 2** in a totally unsupervised way to provide zero-shot personalization to new users on **Site 2**?

As we shall see, the answer is "yes" to both questions.

3 RELATED WORKS

Suggest-as-you-type functionalities are very common in industry products and type-ahead has been a fairly well studied problem in academia in the IR and NLP communities [4]. With the increasing penetration of neural architectures and the successes of "neural" language models (as opposed to more traditional, Markov-style discrete models), recent works have focused on leveraging recurrent neural networks to predict the most likely completion when

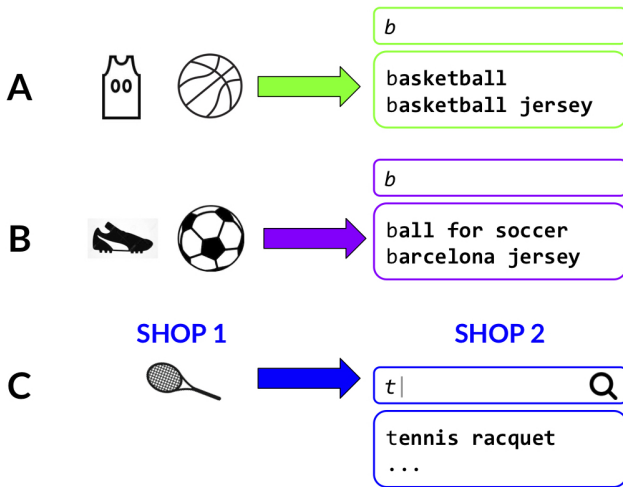


Figure 2: Shopper C is browsing soccer-related products on Site 1, then starts a browsing session on a related site (Site 2). Can we provide personalized type-ahead completions starting from the very first interaction?

shoppers are typing: [12] introduces a character-based language model for query completion; [25] deploys a noisy channel model for prediction as well, but their language model is unconditioned and most of the research is focused on inference speed. The availability of web search (as opposed to *product* search) logs in the public domain has led to work on personalized suggestions with dense vectors mainly based on linguistic features. [8] learns personalized language models by training user embeddings as part of the end-to-end model: our user items are instead learned from catalog features only; they are cheaper to compute and allow an elegant extension of the model to the *cross-shop* scenario as well. [14] uses vector similarity to re-rank suggestions, but it relies on the availability of big query logs and even public datasets (Google News) in the target language. On the commerce side, a recent work from *eBay* [9] embeds previous user queries (through *fastText*) and then re-ranks suggestions accordingly: while we also suggest a retrieve and re-rank strategy, our personalization layer does not require linguistic resources.

A recurring theme in the recent literature in type-ahead is the prediction of “unseen queries” (e.g. [12]): by leveraging neural models, it is argued that type-ahead prediction can be effectively seen as a real-time “language generation” task. From a product perspective, we disagree that “unseen queries” are an important test case for commerce search, let alone the most important one: the final goal for suggesting queries to shoppers is not improving completion click-through rate *per se*, but improving conversion down the line. Since type-ahead predictions are linked to conversion events through the search engine behavior, even a “mind-reading” type-ahead technology - one that always guesses the query intended by the user - is by itself no guarantee of higher conversions (as many “OR”-configured full-text engines will see their performance degrading by adding keywords). In the absence of business KPIs (i.e. conversion rate), we maintain that benchmarking models on

Table 2: Descriptive statistics for data collected for Site 1 and Site 2.

Website	Products	Click events	Queries (type)
Site 1	9002	944643	21488
Site 2	12918	983468	30252

“unseen queries” is not particularly significant when making strategic product decisions (see Section 7 for the role of *offline* language generation in our roadmap).

The literature on language models is extremely vast: even just focusing on neural language models ([1], [19], [28]) - or just conditional neural language models ([10], [11]) -, the literature is growing fast. In digital commerce, the concentration of innovation in a few mono-brand players may have obscured the importance of transfer learning for a vast portion of the market: recently, [2] was the first research to focus on aligning product embeddings and publish proprietary data.

To our knowledge, the current research is the first to explore unsupervised personalization in NLP tasks across digital shops and to provide quantitative evidence and qualitative support that the proposed techniques can successfully address this important business scenario.

4 DATASET

The target dataset comes from a sample of 3 months of historical data for two digital shops, **Site 1** and **Site 2**, two mid-size websites (revenues between 10 and 100 million dollars a year) in the sport apparel space. Collected data includes:

- *Consumer data*: data is collected through a *web pixel* in compliance with existing legislation [24]; for the scope of *this* work, we retrieve product impressions from anonymized shoppers; each impression event contains the *SKU* of the product, that is the product unique identifier according to a shop catalog.
- *Image data*: catalogs from **Site 1** and **Site 2** contain one or more images for each product; images are relatively high-quality (typically 1200x1200), and they are copied from the public URL to our own infrastructure for further processing.
- *Type-ahead logs*: data is collected through a type-ahead API that records what anonymized users are typing (each keystroke), what the user sees in the drop-down menu and what, if anything, gets clicked.
- *Search logs*: data is collected through a search API that records what anonymized users are searching - either through type-ahead or spontaneously.

Descriptive statistics for the final dataset can be found in Table 2; Table 3 shows an anonymized session comprising all types of events; interested practitioners can find additional details on data ingestion and data processing in [2]. As a final remark, **Site 1** and **Site 2** have similarly high bounce rates (approximately 50%) and low ratios of recurring users (<8% of customers have 3 or more visits in 12 months), confirming the business importance of being able to produce session vectors *with minimal information about the user and as early as possible* in the shopping journey. Since

Table 3: A sample session for anonymous user browsing snowboards on Site 1.

Timestamp	Session Id	Event Type	Data
1575207599232	0002bf6d...	view	SKU=0206395
1575207647306	0002bf6d...	suggest	q=d
1575207647718	0002bf6d...	suggest	q=dr
1575207651617	0002bf6d...	search	q=drake got

Site 1 and **Site 2** are independent shops, they are a perfect limit case to test the robustness of *cross-shop* personalization without making any strong assumption of similarity in catalog structure and product taxonomy: it is therefore striking that, without any formal tie between the shops, almost 3% of all sessions in **Site 1** for the test month have a companion session from the same users on **Site 2** (the same metric for brands in within a single group is significantly higher).

5 TYPE-AHEAD MODELS

As a starting point, we assume that a "naive" unconditional model for type-ahead is given by the standard *noisy channel model* (NCM) [25]. Given a set of candidates queries q_1, q_2, \dots, q_n and typed prefix t , we score candidate q according to the Bayes formula:

$$P(q|t) \propto P(q) * P(t|q) \quad (\text{NCM})$$

A big advantage of NCM for commerce search is that it lets us trade-off plausible completions with plausible typos [3] (more than 10% of search queries in the dataset contain typos): $P(q)$ is the "language model", encoding the fact that most users search for "shoes" and not for "sweaters", $P(t|q)$ is the "error model", encoding the fact that "zh" is most likely a misspelling for "sh" and not a genuine prefix. We are making the simplifying but realistic assumption that $P(t|q)$ is dependent on non-contextual factors (device, language, keyboard layout, etc.) that can be pre-calculated, and focus instead on estimating $P(q)$.

Queries in Table 1 show that we should modify $P(q)$ above to $P(q|c)$: while it is theoretically possible to use an n-gram model encoding contextual elements at the start of the query, data sparsity will make the estimates almost relying entirely on smoothing for the vast majority of context/query pairs (a medium-size e-commerce site has more than 20k products and at least 50k pages). We turn instead to dense models, where embedding vectors can be easily plugged in to modify in real-time the unconditioned estimate of the basic model: but what vectors can represent the context well enough?

5.1 Data preparation and session vectors

As the vast majority of shoppers are not recurring, it is crucial to have a representation of unseen users that depends only on the current shopping session. Product images have been shown to perform well (sometimes significantly better than text descriptions) in content-based recommender systems [21]; moreover, exploratory work in cross-shop recommendations confirmed that images could be a powerful feature to aid with transfer learning [2]. It is important to note that relying on images alone for session representation

allows personalization without any assumption on product meta-data or linguistic resources: while *adding* information, if and when available, is a promising path for optimization, the goal of *this* work is to prove that something so minimal like product images are both very scalable and effective (see also Section 7 for additional remarks).

We first extract features from product images as found in **Site 1** and **Site 2** catalogs; a deep convolutional neural network [16] (*fc2* layer from the *vgg16* model) extracts 4096-dimensional vectors from each product image, representing general visual features. PCA is then used to reduce the dimensionality of the vectors to 50 (the 50 components explain approximately 75% of the original variance).

Images can be pre-processed offline as they are available through static catalog files, so no specific deep learning hardware is necessary for data preparation; at session time, as the user is browsing different products, a simple call needs to be made to update a *session cache*, that will retrieve the pre-computed 50-dimensional vector for the product and add it to the list. As explained below, we employ two strategies to encode user session starting from these vectors: first, we take an "average pooling" approach, so that a session with product p_1, p_2, \dots, p_n is represented in the model as the average of the vectors for p_1, p_2, \dots, p_n (averaging vectors to represent user interests is common in the recommendation literature, as for example suggested in [5]); second, when leveraging a full-fledged encoder-decoder architecture, we let the model ingest the vectors for p_1, p_2, \dots, p_n separately and automatically learn how to condition language generation based on them.

5.2 A similarity model

The first improvement to our unconditional model enjoys the benefit of efficient training and easy run-time deployment, at the cost of some simplification on the modelling side. The intuition here is that we can encode the meaning of candidate queries through the vectors of the products that are most frequently associated with them: if the representation is solid, at run-time we re-rank the unconditioned suggestions by the distance in the vector space between the current session vector (representing user latent intent) and candidate queries' vector representation. Using surrounding events to build query representation is also at the heart of the *Search2Vec* [6] model: while philosophically similar, applying the skip-gram model directly to queries is possible only with a huge amount of search sessions; our proposal leverages product interactions as building blocks instead, exploiting the fact that in typical shops most sessions (80% to 90%) are browsing sessions without search events.

In particular, at *training* time, we:

- (1) pre-process images and build a map from products P to their vector representation, $P \mapsto V$;
- (2) retrieve in the historical search data all the products p_1, p_2, \dots, p_n clicked after each of the query candidates in Q , with relative frequency.
- (3) build a map from query candidates to their vector representation, $Q \mapsto V$, by retrieving from $P \mapsto V$ vectors for p_1, p_2, \dots, p_n , and take their weighted average using the frequencies as weights.

At *run-time*, for shopper S , we:

- (1) update the session vector SV in the *session cache* every time S interacts with a product p ;
- (2) retrieve the current session vector SV from the *session cache*;
- (3) retrieve the top N query candidates q_1, q_2, \dots, q_n for query prefix t using the unconditioned language model;
- (4) retrieve for q_1, q_2, \dots, q_n their vector representation from $Q \mapsto V$;
- (5) re-rank q_1, q_2, \dots, q_n by calculating the cosine similarity between each query vector and SV .

5.3 An image-captioning model

Our second improvement to the unconditioned language model has roots in sequence-to-sequence learning: evaluating probabilities of query candidates based on products can be elegantly modeled with an encoder-decoder architecture, as introduced in [18] for machine translation and extended to image captioning in [23]. To encode the current session, we tried the two different strategies outlined in Section 5.1: an "average pooling" version, where the encoder has only one vector, i.e. the session vector resulting from averaging the product vectors in the session, and an "explicit" version, where the encoder takes as input *the sequence* of all product vectors in the session to generate its output, i.e. the encoded session representation.

The model is an encoder-decoder architecture, where the encoder part reads the session information and passes the encoded representation to the decoder, and the decoder is a character-based language model producing strings of text conditioned on the input session representation. The architecture is straightforward: the decoder is built with a single LSTM layer with 128 cells, followed by an output fully-connected layer with a softmax activation. The output dimensionality corresponds to the total number of unique characters in the training data, including the start-of-sequence and end-of-sequence tokens. We set a fixed sequence length for the decoder by taking the maximum length of all queries in the training dataset. During training, the latest cell states of the encoder are passed to the decoder as its initial cell states. At each timestep, we use teacher forcing strategy to pass the target character, offset by one position, as the next input character to the decoder [26].

For training, we use Adam optimizer with an initial learning rate of 0.001 and a decay of 0.00001. Mini-batch of 128 samples per batch is used for training. Cross-entropy loss is used to backpropagate the error and update the weights for our model; training is performed over a maximum of 100 epochs, with early stop and *patience* = 20. Once trained, the model can be used to score a given pair of input context and input query. At run-time, we used the same retrieve-and-rerank strategy mentioned above: we first fetch candidates from the unconditioned model, and then re-rank the top queries using the probabilities of the conditional one. Scoring new queries is done by feeding the context into the encoder and then pass the encoded context, along with an input query, into the decoder. Then, at each timestep, the output of the decoder gives the probability of a target character, given a previous character of the query. Finally, we take the sum of all the output log-probabilities as the final score of the context-query pair. It is mentioned in the literature that such sequence scoring process might favor shorter sequences over longer ones. To overcome this problem, we apply a length-normalization

method by dividing the score of every input query by its length to the power of a real number r (empirically, $r = 0.7$ is a value that works well [27]). From our observations, this length-normalization technique is a simple yet efficient method that allows our model to correct length bias and deliver a better performance.

6 EXPERIMENTS

Type-ahead evaluations on web search datasets typically involve a temporal train/test split, with disjoint time periods to ensure a robust evaluation. In the commerce search case, though, the product context makes it harder to trust pure quantitative measures: completion probabilities at t_n affect search queries frequency, which in turn affect completion probability at t_{n+1} , triggering a "rich get richer" dynamics that vastly overestimates the performance of a frequency-based baseline (the exploration/exploitation dynamics is itself an interesting challenge for live type-ahead systems). While online A/B testing is certainly preferable, sometimes it is not possible to tamper with production systems; for this reason, we chose to have two validation methodologies in place, to combine quantitative metrics in an offline setting and qualitative relevance judgments from humans.

6.1 Within-shop scenario

6.1.1 Quantitative evaluation. The experimental setting follows a standard train/test separation: all the models are trained with data from June to August, and tested on completely unseen events sampled from September. Average results for **Site 1** and **Site 2** over 10 runs, testing 7500 randomly sampled queries each run, are reported in Table 4. We use the *mean reciprocal rank* (**MRR**) as a standard measure from the auto-completion literature; with **MRR@k** we indicate **MRR** as computed with the models returning at most k candidate. In our evaluation, $k = 5$ is picked to be consistent with the target production environments of **Site 1** and **Site 2**:

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i} \quad (1)$$

where rank_i is the position of the first relevant result in the i -th query and Q is the total number of queries. Results are reported for different *seed* length, where *seed* is the characters typed by a user into the search bar, i.e. query prefix, based on which our type-ahead models will be triggered to make suggestions: a seed length of 0 corresponds to the scenario in which the user has just clicked on the search bar, but still has not typed anything; a seed length of 1 is instead testing the suggestions of the given models after one typed character.

The models in the benchmark are the following:

- **Popularity:** standard popularity ranking (e.g. *MPC* in [4]), where $P(q)$ is estimated from empirical frequencies in search logs;
- **Markov:** session is mapped to one out of a set of pre-defined buckets based on the activity type of the products in the sessions (e.g. *tennis*, *soccer*, etc.); type is extracted from the catalog through regular expressions with the help of product experts. The model is a bi-gram based model with Laplace smoothing;

Table 4: MRR@5 at different seed length for models on Site 1 and Site 2.

Shop	Model	Avg. (SD) Seed=0	Avg. (SD) Seed=1
Site 1	Popularity	0.0134 (0.0006)	0.0873 (0.001)
	Markov	0.011 (0.0004)	0.071 (0.001)
	Similarity	0.029 (0.001)	0.121 (0.001)
	Enc-Dec "Avg"	0.0376 (0.003)	0.136 (0.005)
	Enc-Dec "Full"	0.040 (0.002)	0.136 (0.006)
Site 2	Popularity	0.005 (0.0005)	0.0982 (0.002)
	Markov	0.007 (0.0005)	0.081 (0.002)
	Similarity	0.026 (0.002)	0.109 (0.001)
	Enc-Dec "Avg"	0.0374 (0.003)	0.147 (0.007)
	Enc-Dec "Full"	0.0413 (0.003)	0.147 (0.008)





- **Similarity**: cosine similarity re-ranking, as explained in Section 5.2;
- **Enc-Dec "Avg"**: encoder-decoder architecture, using the "average" vector to represent the session, as explained in Section 5.3;
- **Enc-Dec "Full"**: encoder-decoder architecture, using the full list of product vectors to represent the session, as explained in Section 5.3.

Baselines (other than being consolidated industry practices) respect the same data constraints that hold for the proposed models: they don't require linguistic resources and don't make strong assumptions on available product meta-data (see Section 7 for future developments).

Results are robust across sites, showing that dense personalized models significantly outperform popularity and discrete benchmarks: in the most common case of one-character seed, we observe 50% increase in **MRR** between *Popularity* and *Enc-Dec*. Unsurprisingly, the less linguistic information is available (as in the case of predicting queries before any typing), the bigger the gap between the models is. The *Similarity* model, despite its simplicity, is able to capture *some* of the user intent, and it generally provides accuracy in between the baseline and the full deep learning model. It is also interesting to note that the difference between the "average" and the full-sequence models is negligible, perhaps due to the fact that most sessions are relatively short and long-range dependencies may not matter much. To give a sense of the generated predictions, Table 5 collects sample of top queries from non-personalized vs personalized models.

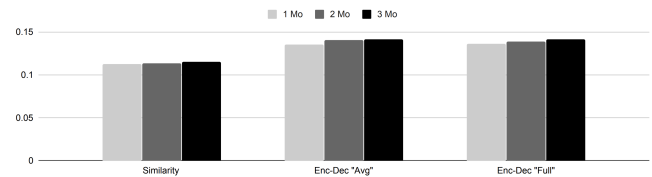
As noted in [20], in the context of web search, not all types of queries benefit from personalization: for some search queries, most users are looking for the same thing; for others, different users want different results even if they use the same wording. To try and capture this effect in the context of product search, we leverage available dense representations instead of using a point-wise metric such as click entropy [20]: a large click entropy means many products are clicked in response to a given query, but it fails to capture the fact that, say, three running shoes are *conceptually closer* in the "product space" than one t-shirt, one soccer ball, one pair of socks in response to the query "nike". In particular, we train *prod2Vec* embeddings for SKUs in the dataset ([2]) and calculate a dispersion

Table 5: Example of type-ahead suggestions (queries are translated) from Site 2, for a given prefix and a given session (represented with a significant product: running shoes, tennis racquets, etc).

Product	Seed	Popularity	Enc-Dec "Avg"
	s	trekking shoes	running shoes
	r	reebok crossfit	tennis racquet
	p	men's polo	men's pants
	t	t-shirt	snowboard

value for each of the query q in the candidate set, based on how tight is the cluster of products associated with it (sum of all distances from the centroid); we then assign all queries in the test set to one of two classes based on dispersion above/below the median value, and evaluate 5 runs of the *Similarity* model for each class. The results greatly confirmed the web search prediction (**MRR@5**: 0.18 vs 0.034 for *highly dispersed vs tightly clustered queries*), confirming that personalized models are at their best dealing with ambiguous cases (i.e. cases in which linguistic information alone is not sufficient to determine product intent). These findings also open interesting lines of work for future refinements, in which a mixed model can be devised to treat differently ambiguous and non-ambiguous query candidates.

On a further note, recent papers have argued that deep learning models in NLP are both inefficient and unfair [17], as the carbon footprint by GPUs is high and cloud costs make developing research papers prohibitively costly for data scientists outside a few tech companies. While we do not take any stance here over these arguments, we *do* think that energy/cost/engineering considerations play a fundamental role in industry settings when calculating the right trade-off for the business: in this spirit, we find important to remark that our cosine similarity model can be trained and deployed successfully without devoted hardware (Appendix A).

**Figure 3: Average MMR@5 for 1/2/3 months of training data for the three dense models.**

Finally, we measured the robustness of the dense methods as the number of samples in the training set changes: the chart in Figure 3 show how the performances change as more data is included in the training set (from 1 to 3 months of data): not surprisingly, the

encoder-decoder methods gain the most in accuracy as training size gets larger, but all dense models are better than baselines after only one month.

6.1.2 *Qualitative evaluation.* To make sure the personalized models are actually capturing a meaningful difference in the "conceptual" space of human speakers, we set up a further validation procedure to avoid relying entirely on the quantitative measure. We recruited 15 native speakers who have no affiliation with *Coveo*, **Site 1** or **Site 2** and whose age ranged between 22 and 45. We presented each of them with a series of *stimuli* such as the one in Figure 4.

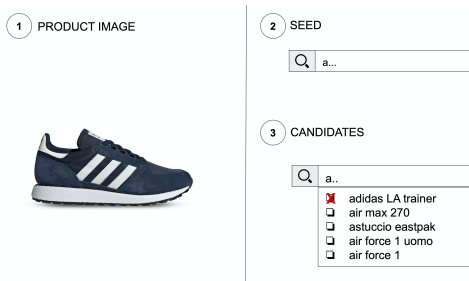


Figure 4: Example of the relevance task: subjects are presented with a product image, a seed and a list of query candidates, and they are asked to mark (red cross) the query that is deemed most appropriate for the product.

The subjects were asked to pick the most relevant completion among 5 candidates, given a commerce product (represented by its image) and a seed; if no suggestion was deemed relevant, subjects were allowed to leave the form blank. The *<product image, seed>* pairs are taken from representative queries extracted from the training set for **Site 2**, for a total of 30 stimuli for each subject; candidate queries are chosen by first retrieving the top 25 candidates from the unconditioned model, and then sampling 5 times without replacement.

By collecting relevance judgment from native speakers *outside* the suggest-and-search loop, our prediction is two-fold: first, dense model should be confirmed to be relevant since conditional models are meant to capture important aspects of the semantic similarity between products in session and queries; second, the performance difference in **MRR** between the baseline and dense models should be higher, since the user study is meant to eliminate the popularity bias implicit in the search logs. After collecting the relevance judgments, we test *Popularity vs Cosine Similarity*, calculating the **MRR** over this new test set: **MRR@5** for *Cosine Similarity* represents a 34.7% increase over the baseline (0.31 vs 0.23), which confirms both predictions; in particular, the purely quantitative evaluation for *Site 2* showed only a 10.99% increase over the baseline (0.0982 vs 0.109). While the present user study was limited in both sample size and scope of semantic intuitions, we believe it provides supporting evidence to the main quantitative evaluation and it could potentially play even bigger roles in understanding specific strength and weaknesses of NLP personalization across different scenarios. For this reason, we look forward expanding this protocol to a larger user study for further iterations of this work.

Table 6: MRR@5 (seed = 1) for shoppers browsing on two shops.

Model	MRR@5 1->2	MRR@5 2->1
Popularity	0.048	0.086
Cross-shop Similarity	0.060	0.093
Cross-shop Enc-Dec	0.083	0.103
Within-shop Similarity	0.096	0.127

6.2 Cross-shop scenario

To obtain a solution to the “zero-shot” personalization challenge in Figure 2, we need to map a session vector for **Site 1** to a session vector for **Site 2**: while product images obviously differ between categories *within* shops and even more *across* shops, the pre-processing pipeline is leveraging abstract features as extracted from a general purposes convolutional neural network that is expected to generalize to many types of objects, photographed under very different conditions. Since product images live all in this general and noise-tolerant “image space”, our hypothesis is that zero-shot personalization can be achieved by combining the vectorization component of a model trained on **Site 1** with the language model trained on **Site 2**.

To test the hypothesis, we run quantitative benchmarks on the cross-shop portion of our dataset. Results for the cross-shop personalization scenario are reported in Table 6, comparing a popularity model where shoppers landing on a new website receive a non-personalized prediction with two personalized models (*Similarity* and *Enc-Dec "Avg"*) injecting a session vector computed from the *previous* site.

The test set is composed by sessions with $n > 0$ products viewed on **Site 1** and then a matching session, the same day, on **Site 2** with $n > 0$ products viewed and one search query issued (we require products to be present on the target site as well to allow for a within-site comparison). Quantitative evaluation for the test period is done on all the 2137 sessions matching the above conditions.

By transferring the session vector between sites, we obtain a significant increase for **MRR@5** over an industry benchmark that treats the incoming shopper as new. To show how hard the task is, we include as a “conceptual” upper bound a *Site Similarity* model, which is a model trained on the target site and using same site context on the test set: it is remarkable that the intent is transferred so well between sites that the best dense model with transfer learning is very close in performance to a same-site model. While the personalization accuracy *across sites* is indeed lower than *within shop*, the main concern of *this* work is proving that transferred intent is significantly better than assuming every shopper is new on the target site; for online shops making close to 100 million dollars in revenue/year or more, capturing the interest of even a small percentage of the portion of non-recurring users may provide significant business benefits.

We also tested the hypothesis that cross-shop sessions closer in time would further improve accuracy for the transfer learning model: if we require the shopper to complete the sessions on the two shops in less than one hour, the gap between unconditioned vs conditional model widens (0.48 for *Popularity* vs 0.72 for *Similarity*

from **Shop 1** to **Shop 2**), but test set size becomes too small to make definite conclusions on this specific hypothesis - we look forward repeating the quantitative evaluations when more cross-shop data becomes available.

It is worth highlighting that by using the shared image space, we do not need to change our language model or re-train it: models for **Site 1** and **Site 2** can be trained and deployed independently; the only change required to unlock zero-shot personalization in a production environment is that, in the absence of history on the target shop, the language model gets injected with a “transferred” session vector.

7 CONCLUSION AND FUTURE WORK

*This work successfully shows that it is possible to leverage easy-to-compute image features to obtain fast and scalable dense representations for commerce products. Starting from an unconditioned noisy channel, we show how to inject personalization with session vectors. The effectiveness of the proposed strategies has been benchmarked with quantitative measures supplemented by a qualitative study, designed to obtain a more nuanced evaluation of the semantic quality of the personalized rankings. Prompted by real industry use cases, we generalized the models to the “cross-shop” scenario, in which shoppers move across similar websites and personalization need to be provided by transferring learned intent from one shop to another. To the best of our knowledge, this is the first work to explicitly address both *within* shop and *across* shops scenarios in auto-completion personalization.*

It is important to remark that the proposed strategy enjoys several advantages over alternative proposals:

- personalization features - i.e. image vectors - are easy to compute, do not need frequent updates (as product images change rarely) and do not require data from the user except within-session interactions: this makes the model very effective for mid-size shops where the majority of users is not recurring and personalization needs to be achieved with as little data as possible;
- our methods do not require catalogs to be particularly accurate in their description, or comparable in their structure: as all digital shops have images for products, the methods presented enjoy widespread, immediate applicability in a retail industry where data quality is not always ideal;
- our methods do not require advance tracking in place to start, making it ideal in integration scenarios in which historical data tracking is incomplete;
- our methods are privacy friendly, as they do not require storing browsing histories for all shoppers: the data in the session cache can expire automatically after most sessions as long as some are kept to continue with model training. At a time of increasing concern for data regulations [24], being able to provide personalized experiences without storing long-term behavioral data on specific users may be an important feature for players in the industry;
- our methods do not require language resources, which is often a problem for mid-size, multi-lingual commerce sites: on the one hand, linguistic resources (say, word vectors [13]) are often not available for all languages; on the other, most

shops do not have enough textual data to train effective vectors themselves;

- our methods provide a principled way to transfer learning across sites, which is one of the biggest challenges for e-commerce service providers and a massive opportunity for multi-brand retailers, which often need to transfer learning from data-rich stores to “zero-data” stores;
- finally, the methods allow for a step-wise implementation process, which allows for incremental change to standard industry type-ahead APIs (see Appendix A for engineering details).

While our results are very promising, our research opened up many possibilities for further improvement, both from a product and research standpoint. The deployment of full-fledged in-session personalization on all NLP touchpoints is one of the most important items in *Coveo*’s product roadmap: in particular, our multi-brand groups will be involved in all the product choices (data ingestion, real-time processing, etc.) needed to deploy personalization at scale. We are also investing heavily in improving engineering and model serving, making sure the uplift in performances and elegance gained with dense models is sustainable at scale in a fast growing organization.

Our research roadmap on type-ahead mainly focuses on two themes. For reasons mentioned in Section 3, we are skeptical of unconstrained real-time natural language generation, potentially resulting in unseen queries with low business value (i.e. conversion rate) and unnecessarily complex serving infrastructure. However, we do believe a big goal of type-ahead is helping *discovering* portions of the product space the users may not even know exist. To that extent, natural language generation as performed *offline* at indexing time is a powerful technique to increase the number of query candidates in a controllable way. A second point is obviously the conditioning process, as product images are easy to compute, but their representation potential may be augmented by other unsupervised representations, ranging from behavioral-based product embeddings [2], to page embeddings (i.e. not necessarily product pages), to longer-term user information (when available): *this work proved that image-based deep architectures significantly improve performances of type-ahead services, however it is still an open question which combination of dense representations and deep architectures exactly perform the best. We leave this point to future research.*

Finally, it is our deep conviction that transfer learning is a core component of any commerce solution with the ambition of serving hundreds-to-thousands of customers in a global market: for this reason, the understanding of “cross-shop” behavior is an active area of research in our lab, from recommendation [2] to several NLP tasks.

ACKNOWLEDGMENTS

Thanks to Luca Bigon for support on data ingestion and engineering tooling needed for the project; thanks to three anonymous reviewers, whose feedback greatly improved the paper. Finally, thanks to our clients, who are instrumental in the success of the company and have been very receptive to the possibilities opened by A.I. in retail.

REFERENCES

[1] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2001. A Neural Probabilistic Language Model. In *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp (Eds.). MIT Press, 932–938. <http://papers.nips.cc/paper/1839-a-neural-probabilistic-language-model.pdf>

[2] Federico Bianchi, Luca Bigon, Jacopo Tagliabue, and Bingqing Yu. 2020. Fantastic Embeddings and How to Align Them: Transfer Learning Across Shops Through Dense Product Representations. *Under review* (2020).

[3] Eric Brill and Robert C. Moore. 2000. An Improved Error Model for Noisy Channel Spelling Correction. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Hong Kong, 286–293. <https://doi.org/10.3115/1075218.1075255>

[4] Fei Cai and Maarten de Rijke. 2016. *A Survey of Query Auto Completion in Information Retrieval*. Now Publishers Inc., Hanover, MA, USA.

[5] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. Association for Computing Machinery, New York, NY, USA, 191–198. <https://doi.org/10.1145/2959100.2959190>

[6] Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, Fabrizio Silvestri, Ricardo Baeza-Yates, Andrew Feng, Erik Ordentlich, Lee Yang, and Gavin Owens. 2016. Scalable Semantic Matching of Queries to Ads in Sponsored Search Advertising. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '16)*. Association for Computing Machinery, New York, NY, USA, 375–384. <https://doi.org/10.1145/2911451.2911538>

[7] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. 2015. E-commerce in Your Inbox: Product Recommendations at Scale. In *Proceedings of KDD '15*. <https://doi.org/10.1145/2783258.2788627>

[8] Aaron Jaech and Mari Ostendorf. 2018. Personalized Language Model for Query Auto-Completion. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Melbourne, Australia, 700–705. <https://doi.org/10.18653/v1/P18-2111>

[9] Manojkumar Rangasamy Kannadasan and Grigor Aslanyan. 2019. Personalized Query Auto-Completion Through a Lightweight Representation of the User Context. *CoRR* abs/1905.01386 (2019). [arXiv:1905.01386](http://arxiv.org/abs/1905.01386) <http://arxiv.org/abs/1905.01386>

[10] Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A Conditional Transformer Language Model for Controllable Generation. *ArXiv* abs/1909.05858 (2019).

[11] Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. *2012 IEEE Spoken Language Technology Workshop (SLT)* (2012), 234–239.

[12] Dae Hoon Park and Rikio Chiba. 2017. A Neural Language Model for Query Auto-Completion. <https://doi.org/10.1145/3077136.3080758>

[13] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 1532–1543. <https://doi.org/10.3115/v1/D14-1162>

[14] Taihua Shao, Honghui Chen, and Wanyu Chen. 2018. Query Auto-Completion Based on Word2vec Semantic Similarity. *Journal of Physics: Conference Series* 1004 (apr 2018), 012018. <https://doi.org/10.1088/1742-6596/1004/1/012018>

[15] SimilarWeb. 2019. *Top sites ranking for E-commerce And Shopping in the world*. Retrieved December 23, 2019 from <https://www.similarweb.com/top-websites/category/e-commerce-and-shopping>

[16] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.

[17] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and Policy Considerations for Deep Learning in NLP. In *ACL*.

[18] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *NIPS*.

[19] L. Burget J. Černocký T. Mikolov, M. Karafiát and S. Khudanpur. 2010. Recurrent neural network based language model. *INTERSPEECH-2010* (2010), 1045–10489.

[20] Jaime Teevan, Susan T. Dumais, and Daniel J. Liebling. 2008. To Personalize or Not to Personalize: Modeling Queries with Variation in User Intent. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '08)*. Association for Computing Machinery, New York, NY, USA, 163–170. <https://doi.org/10.1145/1390334.1390364>

[21] Hessel Tuinhof, Clemens Pirker, and Markus Haltmeier. 2019. Image-Based Fashion Product Recommendation with Deep Learning. In *Machine Learning, Optimization, and Data Science*, Giuseppe Nicosia, Panos Pardalos, Giovanni Giuffrida, Renato Umerton, and Vincenzo Sciacca (Eds.). Springer International Publishing, Cham, 472–481.

[22] Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2018. Meta-Prod2Vec - Product Embeddings Using Side-Information for Recommendation. In *Proceedings of RecSys '16*. <https://doi.org/citation.cfm?doi=2959100.2959160>

[23] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2014. Show and tell: A neural image caption generator. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014), 3156–3164.

[24] Paul Voigt and Axel von dem Bussche. 2017. *The EU General Data Protection Regulation (GDPR): A Practical Guide* (1st ed.). Springer Publishing Company, Incorporated.

[25] Po-Wei Wang, Huan Zhang, Vijai Mohan, Inderjit S. Dhillon, and J. Zico Kolter. 2018. Realtime Query Completion via Deep Language Models. In *eCOM@SIGIR (CEUR Workshop Proceedings)*, Jon Degenhardt, Giuseppe Di Fabbriozio, Surya Kallumadi, Mohit Kumar, Andrew Trotman, Yiu-Chang Lin, and Huasha Zhao (Eds.), Vol. 2319. CEUR-WS.org. <http://dblp.uni-trier.de/db/conf/sigir/ecom2018.html#WangZMDK18>

[26] Ronald J. Williams and David Zipser. 1989. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks.

[27] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *ArXiv* abs/1609.08144 (2016).

[28] Ziang Xie. 2017. Neural Text Generation: A Practical Guide. *ArXiv* abs/1711.09534 (2017).

A ARCHITECTURAL NOTES

As specified when formulating NCM, we assume as a baseline implementation a “naive” unconditioned language model based on popularity and deploy a retrieve-and-re-rank strategy, first retrieving suitable candidates and then scoring them with the *conditional* language model we learned from the data. Assuming we can generate the list of candidate queries in advance (see also note on query generation in Section 7), our starting implementation is a trie-based architecture leveraging the fact that both the language model and the error model can be estimated offline when building the index. An example of this data structure for indexing query candidate *shoes* is depicted in Figure 5:

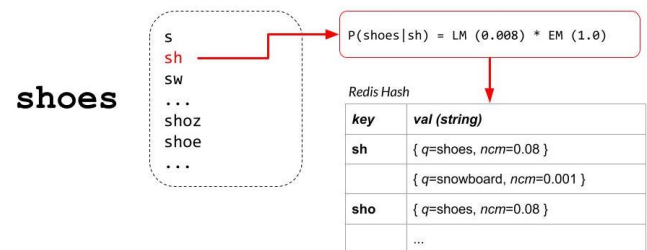


Figure 5: Indexing *shoes* in a trie built on top of an hash-map, pre-calculating at indexing time the unconditioned probabilities for all the possible completions of a given prefix.

At run-time, we retrieve from the trie the top N completions for the given prefix, since they are stored in descending probability order (this data structure allows us to trade off space for time, resulting in constant query time thanks to hash computational complexity¹).

We can progressively enhance this setup by introducing in steps our two personalization models. First, we can augment the indexing process *without* deploying expensive hardware with another offline process - image vector calculations - to prepare for the similarity-based personalization; we first retrieve for *shoes* images of products

¹<https://redis.io/commands/hget>

clicked by users issuing the query, run them through the CNN and store the resulting vector in the same JSON representing the property of the completions in the Redis hash (Figure 6). At run-time, we retrieve the top $N * k$ completions and re-rank them based on the distance between their vectors representation and the current session vector, as retrieved by a session cache: to maintain the *session cache*, the easiest solution would be to adopt a lightweight memory store updated at each product view.

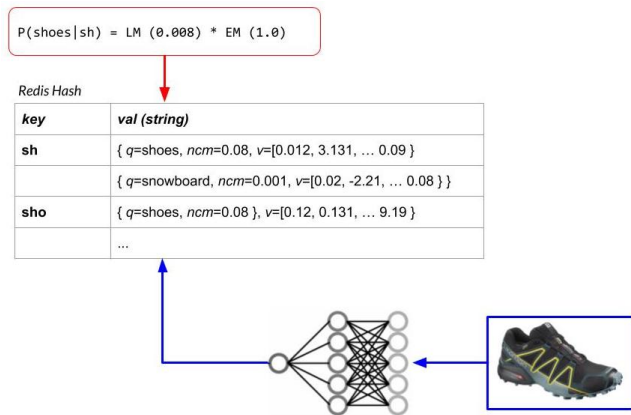


Figure 6: When dense vectors are introduced, indexing shoes in a trie structure requires merging historical search data with the results of the image processing pipeline.

We maintain that the simplicity of this extension is an argument in favor of the similarity model, but we would still like to find ways to extend the architecture to accommodate the encoder-decoder model with minimal disruption. In particular, we focus on a solution that allows us to decouple deep learning models for personalization from query time constraints - a solution that degrades gracefully in case of any problem to the deep learning serving layer. Our proposal for an "hybrid system" of this kind is depicted in Figure 7.

Figure 7 shows three timelines - *User*, *Shop*, *Model* - and some basic *Infrastructure* resources at the bottom. Going from left-to-right, we follow the user journey through the website:

- at each product view, the shop *S* takes care of synchronously updating the cache;
- each cache update triggers an asynchronous model run (*M* timeline) to estimate the conditional probabilities for the top *U* queries (where *U* is a much greater number than what is usually displayed to the user on the website), which are stored in a temporary hash in the query completion database;
- when the shopper finally types "s" in the search bar and the website needs to retrieve suggestion, it retrieves the top $N * k$ completions, but re-rank them with the conditional probabilities for the queries as stored at the last model update.

In other words, the model layer acts in a sort of "best effort" mode, trying to update as fast as possible query candidates probabilities to provide more accurate suggestions, knowing well that not every candidate will be ranked correctly, but knowing that most candidates will be without corrupting in any way the user experience on the website. An additional benefit of this hybrid approach is that

the deep learning infrastructure can be introduced gently first and with minimal investment, and optimize later (both on hardware and software side) when ROI is more easily measured.

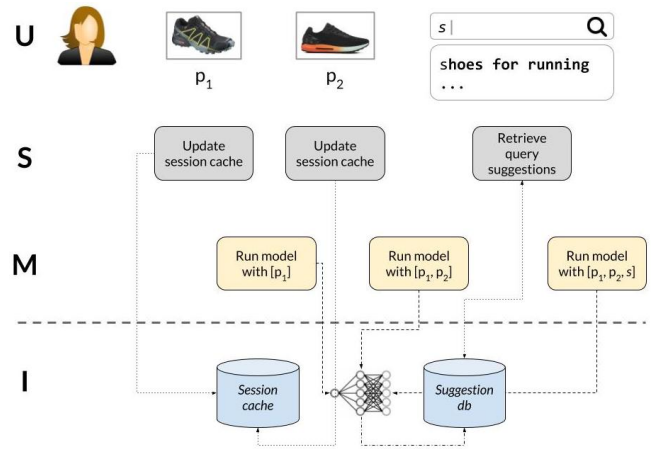


Figure 7: Shopper, website and model timeline together in a hybrid system that is achieving personalization in "a best effort" mode: as the shopper moves on the target domain, neural networks behind the scene adjust probabilities for candidate queries, decoupling run-time performance of the API from the personalization component.