(Vector) Space is *Not* the Final Frontier: Product Search as Program Synthesis

Jacopo Tagliabue
jacopo.tagliabue@nyu.edu
New York University, Bauplan
New York City, NY, USA

Ciro Greco
ciro.greco@bauplanlabs.com
Bauplan
New York City, NY, USA

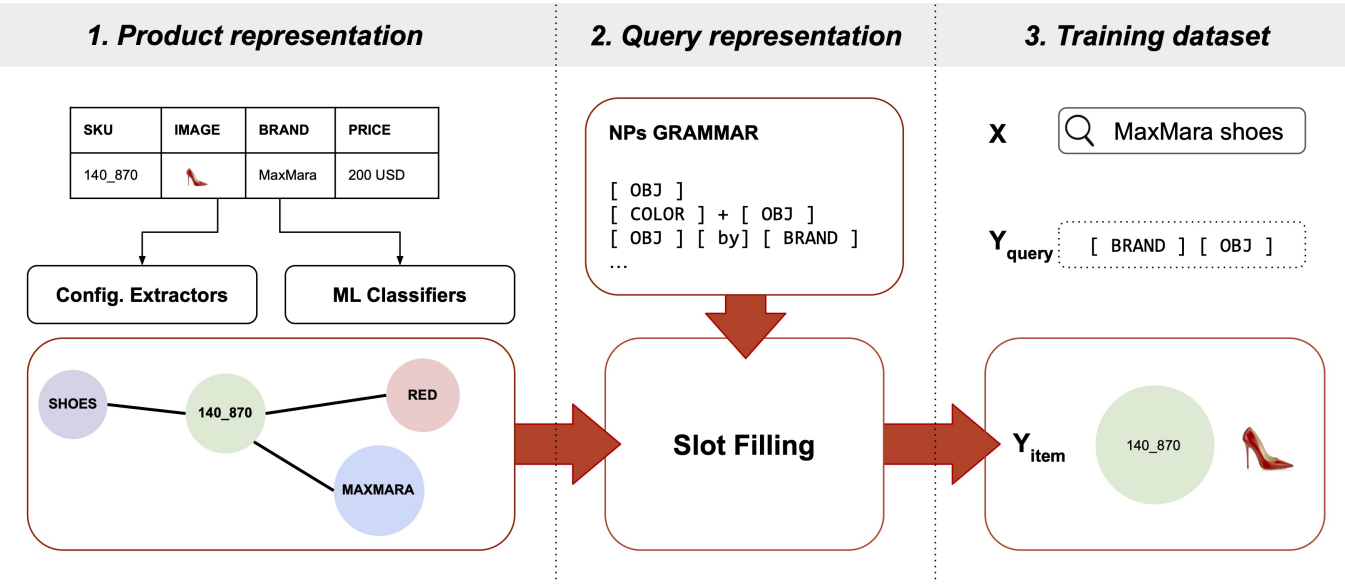

Figure 1: How to build a query parser with no manual annotations (Section 4). (1): extractor for product features from the catalog; (2): a grammar covering common logical forms; (3) dataset generated with queries and matching semantic form, to train a parser for runtime query analysis.

ABSTRACT

As ecommerce continues growing, huge investments in ML and NLP for Information Retrieval are following. While the vector space model dominated retrieval modelling in product search – even as vectorization itself greatly changed with the advent of deep learning –, our position paper argues in a contrarian fashion that program synthesis provides significant advantages for many queries and a significant number of players in the market. We detail the industry significance of the proposed approach, sketch implementation details, and address common objections drawing from our experience building a similar system at *Tooso*.

KEYWORDS

product search, semantic parsing, program synthesis, large language models

SIGIR eCom'23, July 27, 2023, Taipei, Taiwan

ACM ISBN 978-1-4503-XXXX-X/18/06.

ACM Reference Format:
Jacopo Tagliabue and Ciro Greco. 2023. (Vector) Space is *Not* the Final Frontier: Product Search as Program Synthesis. In *Proceedings of ACM SIGIR Workshop on eCommerce (SIGIR eCom'23)*. ACM, New York, NY, USA, 7 pages.

1 INTRODUCTION

> "Now, like all great plans, my strategy is so simple an idiot could have devised it" – *Zapp Brannigan*

The explosive growth of ecommerce [16] brought equally impressive innovation in Information Retrieval (IR) [1], with *product search* now to representing 30% to 60% of total online revenues [2, 14, 15]. Building on decades of literature in web and document retrieval, product search is typically modelled as a two-step process: *candidate selection* (*retrieval* [40]) and *re-ranking* [13, 30, 33, 36]. The most widespread model for retrieval is the *vector space model* (VSM) [20, 26, 50], according to which relevance is approximated by the distance between a query vector and a product vector in a suitable space. Even as deep learning drastically altered vectorization [35], it did not call into question the tenets of the VSM, or the idea that re-ranking is needed to push down the page irrelevant items wrongfully retrieved [43, 55]. It is important to remember that most real-world search engines leverage VSM in one form or another:

sparse BM25 retrieval in Elasticsearch may be implemented very differently from dense retrieval on Redis Vector Search[1], but they all share the core idea of VSM. Namely, that retrieval is fundamentally approximated by distance in a vector space.

We argue that *program synthesis through semantic parsing* provides a principled and viable alternative to VSM for product search. In this perspective, search queries are (informal) instructions for knowledge bases, as opposed to points in a vector space [2]. We shall defend two main claims:

(1) VSM is an *indirect* representation of *meaning* that is necessary for large unstructured documents, such as those in web search; however, under different circumstances, where search queries are interpreted against product catalogs, *direct* representation is feasible and useful;
(2) explicit representations unlock a powerful search experience where formal inferences can be made to improve retrieval, while ranking is used as a device for personalization.

Historically, ecommerce tech has been focusing mostly on the challenges of big players, while a larger market share represented by mid-to-large websites has been neglected [44]. While we recognize the intrinsic limits of *position* papers, we believe our contrarian argument will benefit from the freedom allowed by this format. Our arguments proceed as follows: we first establish some empirical facts about ecommerce search at the "Reasonable Scale"; we then showcase the virtues of program synthesis, *assuming* a semantic oracle. Finally, we show how such a system can actually be built.

We believe this work to be valuable for a broad set of practitioners, solving specific use cases in this segment of the market or working on SaaS solutions[3]. Even if most of the arguments we present are theoretical, these ideas have been successfully implemented in a company before (*Tooso*), and played an important role in its acquisition by a public market leader (*TSX:CVO*)[4].

2 AN INDUSTRY PERSPECTIVE

> "Hooray! A happy ending for the rich people." – *Dr. Zoidberg*

While the idiosyncrasies of product search have been partially documented before [10, 52], most ecommerce systems are still designed from the same building blocks as document search: VSM for retrieval, Machine Learning (ML) for re-ranking using all types of signals. In our experience, the farther you go from planetary scale retailers, the less product search will resemble web search.

Because digital transformation is consistently taking place in the retail industry, most ecommerce search systems are now deployed outside of Big Tech Retailers. We are going to describe the mid-long tail of ecommerce implementations as the "Reasonable Scale" (RSc)

[1] https://redis.io/docs/stack/search/reference/vectors/

[2] As we explain below (Fig. 4, our approach is to parse a search query to an intermediate semantic representation, and then translates the latter into a program, handling the shopping query "as if it were instructions"; program synthesis may also be construed directly from natural language [4]. We will refer to parsing and synthesis somehow liberally below, since it's clear how to move from one to the other.

[3] As a business context about this blooming industry, Algolia and Bloomreach raised >USD200M each in venture money in the last few years [9, 51], and Coveo raised >CAD200M at IPO [32].

[4] While most of these ideas have been developed in 2017-2019, we have updated our arguments to reflect the most recent advancements in the field.

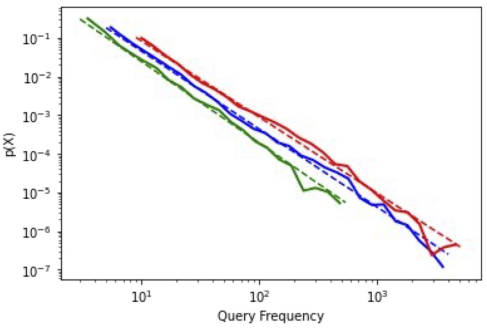

Figure 2: Anonymized query frequency distribution (and fitted power-law PDF) on a log-log plot for three RSc shops in the literature [8, 38, 48].

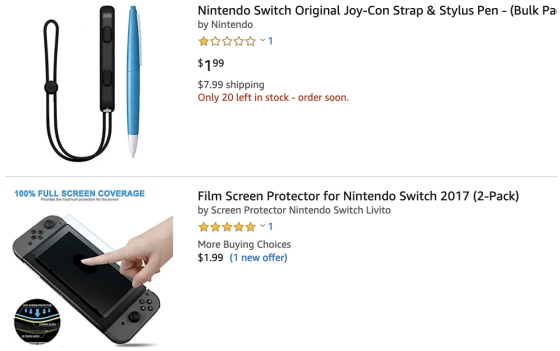

Figure 3: Price re-ordering on *Amazon.com*, showing degrading relevance in the result set when querying for "nintendo switch", and then re-ranking based on price.

[18, 34, 46, 47]. While RSc is intended to be a loose concept [44], practitioners typically know it when they see it [5].

A number of strategies need to be different at RSc. For instance, instead of several millions of SKUs, RSc shops may have 10K to 100K products and still make >100M USD in yearly revenues. Queries on inventories of this size can easily have result sets of 10 golden items. In this context, no re-ranking strategy will be able to hide irrelevant products from the user: for the typical strategy of hiding results in page two[5] to work, there should be a page *two* to begin with. Even as inventory grows, VSM may go against shoppers' preference: for price-sensitive items, users often sort results by price [49]. When this happens, sub-optimal candidate selection can hurt the experience (Fig. 3)(see also the cases discussed in [27] with regard to prices and sizes).

To paint a more quantitative picture of the RSc, we can leverage our unique and privileged position as SaaS practitioners with access to dozens of different real-world deployments. In particular, there are two main facts that turn out to be crucial for our approach (Section 4):

(1) product search mostly deals with short queries in the form of Noun Phrases (NPs) describing entities and properties (e.g. "red shoes" or "Dell laptop") [41]. Query examples from RSc shops can also be found in [56] (Table 1) and [6];

[5] "The best place to hide a dead body is page two of Google."

(2) a small number of queries account for a significant portion of the distribution, making superior relevance for top queries extremely impactful for the overall experience. In the frequency distribution of a month of anonymous query data sampled from three RSc shops in two languages, the top 1-to-5% queries account for *half* of the total individual queries (Fig. 2).

The first observation is important as parsing gets harder with longer queries; the second observation is important as it indicates how to align technological objective with business outcomes – i.e., solving parsing for short queries is a very good place to start.

Taken together, they both re-affirm the peculiarities of product search, but from a novel and unusual angle: interestingly, both facts are *not* true for web or big-scale ecommerce search – as the numbers of users / items get larger and revenues grow into billions, the tail of the query distribution gets both longer and more important. In other words, while the general linguistic behavior for users of *Amazon* or *Facebook* is also be NP-based, the tail is disproportionally more important: the tail is longer, as big catalogs invite a larger set of inputs, and the tail is more valuable, as marginal improvements in rare queries translate in sizable monetary gains. While we believe our approach can be used, under the appropriate circumstances, at any scale, its novelty and impact are more easily noticeable for RSc deployments.

3 SEARCHING WITH AN ORACLE

Originally developed for large documents and long queries, VSM is a useful approximation as it provides a retrieval strategy that avoid *explicitly modelling for meaning*, which has long been thought to be an intractable problem: what would be the logical form [25] of this Wikipedia page[6]? As we argue below, the challenges of explicit representations are eased for product search: on the query side, real-world data shows that NP-like queries are very impactful (Section 2); on the item side, products are remarkably different from long documents: products are well-defined entities, which can be described through a sortal (i.e. the type of object, e.g. "shoes") and few key properties (e.g. color, material, size, brand, price - crucially those more often used by shoppers [6, 7]). In other words, products already come into an IR system as (quasi) *structured* information.

What would a search-as-parsing experience look like? We first sketch the general experience we have in mind through a "parsing oracle" (PO) - i.e. an idealized system that is able to:

- at runtime, return the logical form of a query;
- at indexing time, given a product (as contained in a digital catalog [48]), return its properties.

Under the proposed approach, a query is parsed into a logical form (*parsing*), which is mapped to a machine code to be executed over the target domain (*synthesis*): in Fig. 4 we find lambda expressions and SQL [23], but the proposal is broadly compatible with any explicit formalism. In other words, the *meaning* of "Prada purple shoes" is neither boolean operators over TF-IDF weights, nor a BERT-based embedding, but (something like):

$\lambda x.[Purple(x)\ \&\ Shoes(x)\ \&\ Prada(x)]$.[7]

[6]https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)

[7]With $Purple$, $Shoes$, $Prada$ as predicates of type $Color$, $Sortal$, $Brand$.

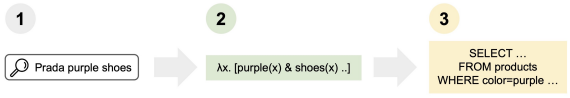

Figure 4: From user query (1) to executable code (3), through semantic parsing (2). Under the assumption that PO exists, queries could be executed as programmatic instructions through their logical form.

Viewing queries as (small) programs to execute has several advantages. First, it provides the ability to apply filters that are already available and show their application to the user – this is often desirable but in a VSM-system it requires an additional module to be trained and maintained. Second, the explicit and easy-to-debug "trace" of the query enables principled fallback strategies. As an illustration, assume the user issues the query "purple shoes", which has no perfect matches. The logical form that (roughly) states "retrieve an object of type shoes, with purple as a color", allows us to reason about the next best thing available, and provide a graceful fallback message (e.g. "we don't have purple shoes, but we thought you could like dark red shoes"[8]. An explicit parse leads us to recognize that different tokens in the query have different psychological importance for the shopper: if the retrieval goal revolves around shoes, the system should retrieve items that are still shoes while never retrieving purple items that are not shoes. In this perspective, parsing both yields the exact linguistic intent and lays down possible compositional fallback strategies. Crucially, fallback strategies can be ML-driven, domain-driven, or heuristic-driven and may change from one deployment to the next: by turning queries into code, we make it easier to incorporate constraints (including probabilistic ones) into an interpretable search plan.

To further appreciate the experience, it is useful to contrast what would have happened under plausible implementations of VSM. Under a sparse vector space, the shopper would typically either get a *No result page*, or – as the opposite extreme – received irrelevant items from an OR expansion: non-shoes that are purple, shoes that are green [9]. Under a dense vector space, retrieval would provide a set of items, but no principled way to cut the set at the right position (when a "near" vector is not near enough?) or explain its choice. Both are open problems [3], and no solution is known, especially given data constraints of the RSc [47].

There is another, subtler, way to appreciate the impact of PO on search especially relevant for SaaS players [45], whose job is to develop solutions deployed on dozens independent shops in several languages and verticals. When you have two shops in the same market (as **Shop A** and **Shop B** below), PO gets you re-usable abstractions. Overlapping parse trees and product properties can help with cold start scenarios: if a model is matching "Adidas" and "Nike" as brands with high affinity, it can be ported to a new shop to boostrap learning (i.e. bootstrapping a new ecommerce without any behavioral data). As an even more extreme form of bootstrapping, learning can be transferred between (similar) languages when appropriate resources exist: while VSM models can make good use of

[8]Note that while IR explanations are often used to improve recommender systems [57], search may benefit from them for similar reasons.

[9]Far from being a theoretical possibility, this is the default experience for all website using out open source tools like Elasticsearch, or non-AI SaaS providers.

multi-language embeddings, the power-law of RSc helps us here as well, as most retailers would do most business in 1-3 languages. [10] Of course, re-imagining search with PO opens up possibilities also outside of the search experience itself: just to mention two obvious ones, finer-grained analytics (both about queries as expressing shoppers' intent, and products, as a collection of human-readable properties), and cross-pollination with data coming in and out of the PIM (Product Information Management).

In this section, we argued that a large portion of the market would benefit from program synthesis through semantic parsing, if such a system existed. We now show how such system can be built.

4 BUILDING A SEMANTIC PARSER

As PO itself has two components – a query and a product parser with a shared domain and interpretation (in the sense of model theory [22]) – how do we bootstrap and scale both? Assuming we use ML to train the parser, the hardest part is obtaining a training set for queries: while (almost) any untrained human can annotate an ecommerce catalog, producing logical forms requires a good deal of work by trained linguists. We will therefore break the problem into pieces, by first assuming we have product representations available to build a training set for query parsing, and then relaxing this assumption.

Fig. 1 showcases the creation of a query dataset for a statistical parser (3) [11], starting from product representations (1) and a small grammar (2): our insight is that, instead of manual annotation, we can programmatically generate golden triples *<query,logical form,SKUs>* by synthesizing *jointly* queries, their logical form, and the result set, leveraging the isomorphism between product representations and logical forms. Moving the annotation problem away from logical form helps us leverage further insights on the peculiarities of the RSc. First, it should be stressed that extracting (most) product features (1, in Fig. 1) is easy: some attributes come already structured, and statistically accurate labels are easy to obtain thanks to methods applicable across shops [11, 21]. In particular, while recent large language models cannot be directly used at runtime [24, 31], they are ideally suited to be a complementary strategy to more traditional methods when it comes to entity extraction (or even as an oracle for offline usage [17], see the Appendix). Product information is also important for other parts of the business, which means labeling can piggyback on independently motivated processes (e.g. PIM).[12]

Second, the peculiarities of query distribution simplify the slot filling component (2, in Fig. 1): even in a SaaS scenario where extreme scalability is paramount, NP queries are easy to generate and then re-use – the queries "ski trousers", "running shoes" and "ski gloves" (mentioned in [56]) share the same logical form. Not only the grammar is simple enough to start, but since the final goal is to parse queries through a model trained on these synthetic NPs, we can err on the side of recall and over-generate (as it will just create training sentences that nobody would use).

[10] Even the fallback strategies mentioned before can be ported: if "sneakers" is fallback for "shoes", the same strategy can be applied any time you have "shoes" available in the parse tree.

[11] The details of the parser are pretty unimportant, as there is substantial evidence that this is a solvable problem with good enough data [53].

[12] Product labeling can also be outsourced with no privacy concerns.

Let's now recap our approach as an actionable list:

(1) at indexing time, extract product representations from the catalog to be indexed in a knowledge base, through heuristics and/or models [11, 37];[13]
(2) build a simple NP-focused grammar, to cover a significant part of the distribution. The process can begin by annotating historical queries with simple logical forms, and then generalize a grammar to simplify those trees. To give a sense of how this would work, we selected **Shop A** and **Shop B**, multi-brand retailers in the apparel industry and catalog size between 10k and 30k SKUs. We manually annotate historical queries to get a sense of what grammar captures user behavior. Few hundreds parses (respectively, 475 and 459) cover 43% and 25% of the entire query distribution for **Shop A** and **Shop B**;
(3) use the product representation and the NP-grammar to generate a training test with synthetic queries and golden parse trees (Fig. 1) – note that it is easy to augment the set of parsable queries through paraphrases [53] or prompting [39];
(4) train a standard parsing model [28, 58] on this dataset;
(5) at runtime, use the parsing model on an incoming query, get the logical form and map it to an executable code for the target knowledge base: retrieve the products, execute fallback strategies if relevant.

This strategy has consequences for two important pieces of the search experience, *re-ranking* and *type-ahead suggestions*. Reranking in VSM is often needed to hide poor results, and may even conflict with relevance objectives: e.g., popular products may sometimes outrank others irrespective of query intent. A structured approach to retrieval allows ranking to be mostly about personalization: given a relevant result set, which of the following "purple shoes" is best for this shopper (based on several real-time and historical ranking signals)? Conversely, ranking rules – both manual and learned – can be applied on a *ceteris paribus* level: only if two items are equally relevant, popularity can influence their ranking. Query suggestions are known to be important for a good search UX [49]: synthetic queries (Fig. 1) could be used to suggest new and cold query types, as well as familiarize shoppers with the capability of the parser; for example, suggesting "blue shoes under 100 USD" would gradually educate users in using the search bar better.

5 LIMITATIONS AND ANSWERS TO COMMON CONCERNS

5.1 Vectors strike back

The explosion of NLP-capabilities in recent years have established beyond any reasonable doubt the virtues of distributional semantics [29]: it may therefore seem strange to defend program synthesis for IR use cases. The quality of the vectorized representations for queries and products increased dramatically (including exciting possibilities such as multi-modal understanding [12]), but the problem with VLM is still present even in the most sophisticated retailers: as we observe in the result set in Fig. 3, the query "nintendo switch" is retrieving *pens*. While it would be tempting to dismiss this as an

[13] We refer the readers to the Appendix for more details.

artefact or an anecdote, it is on the contrary an essential component of VSM: if relevance is distance in a vector space, there is no *cut-off* establishing when far is *too* far. If we compare the result set to the typical response we would get from a human assistant[14], it is clear the shared meaning of "nintendo switch" is very different. For almost-web-scale catalogs, vector search is pragmatically an effective strategy, as the "very close" products for most queries are enough to fill the first few result pages; for smaller catalogs, however, the perceived relevance may quickly degrade and the VSM approach has no principled countermeasure.

As we discuss below, better vector representations are an essential component of *any* search engine, and NLP breakthroughs are a welcome addition to the toolkit of any shop. However, treating relevance solely as a distance calculation is an approximation, and should be recognized as such: when we switch our attention from lexically-driven to compositionally-driven use cases, how much value can we now unlock?

5.2 Parsing vs rewriting

Parsing is hardly the only query processing technique available to RSc shops: for example, query rewriting is a popular approach to bridge the gap between the user's intent ("red Nike sneakers") and inventory (burgundy Adidas shoes). However, it is important to realize that the concerns of parsing and rewriting modules are distinct, and possibly complementary: you can rewrite "sneakers" into "shoes" before parsing it into an object type, but rewriting by itself does not challenge the fundamental assumption of VSM – effective rewriting may improve recall, but does not unlock any of the relevance benefit that parsing provides (Section 3). From an engineering perspective, it's easy to see how a rewriting module could leave completely untouched the retrieval machinery of VSM, while parsing requires re-thinking the strategy entirely.

Moreover, a crucial component of our proposal is the "zero-shot" adaptation obtained through the loose isomorphism between products in a graph and grammars: since parsing is built through product understanding, not explicit or implicit behavioral supervision, its sample efficiency makes it ideal for RSc shops and horizontal scalability (see below); on the other side, modern NLP-based rewriting through behavioral supervision [54] is better suited for big retailers.[15]

5.3 Vertical vs horizontal scaling

When thinking about "scalable" engineering, we think of diminishing marginal effort as we "scale" along an important dimension. Since most IR is done at Big Tech scale, the implicit notion of scalability is the B2C one: as a target shop grows in inventory and traffic, the long tail of queries will expand and rare events become more important (Section 2). In this regime, data-driven approaches are *scalable*: the more traffic, the more data, so statistical generalization is a promising path to diminishing marginal effort – how hard is to satisfy this shopper's intent, given we have seen already k million of them?

As we hinted in *this* work, there is another concept of scalability in IR, which becomes evident for B2B scenarios: if our system is used across multiple RSc shops, the marginal cost that will dominate the business is *deployment* cost – how hard is to get a new shop online, given we put online k already? The synthesis approach we championed has been developed mainly targeting this second notion: if the marginal cost of tagging catalogs is diminishing (see the *Appendix*), the cost of understanding queries on newer shops diminishes as well, *irrespective* of how much traffic they get. While emphasis has been put on synthesis as the actual implementation mechanism for our strategy, the broader, and perhaps novel insight, is that query performance is (in certain cases) a by-product of product understanding and linguistic knowledge, both of which are more scalable than practitioners typically realize.

5.4 Parser fragility

A critical point that has not been addressed is the "fragility" of parsing-first strategies: since no parsing model would be perfect, what should we do when it fails? In our experience, the most natural architecture is a two-tier system, such that, if parsing or program execution fail, the system would resort to a traditional VSM strategy (e.g. a sparse / dense vector-based retrieval). Considering the speed of an ML parser, we pay a tiny latency tax for the above mentioned benefits. When it comes to deployment, our recommendation is to use program synthesis on top of a basic VSM retrieval, not as a replacement; philosophically however, our position remains that VSM is an *approximation* to relevance, and should be treated as such.

6 CONCLUSION

Motivated by query distributions and industry constraints, we argued that program synthesis (through semantic parsing) is a feasible path for a better search experience at RSc, compared to VSM alone as a relevance model. We showed that the usual worries associated with explicit meaning representations are unwarranted, and maintained that the key insight to a novel view on search is the "isomorphic" structure of (parsed) queries and product structure.

The representation dichotomy *explicit-but-annotation-heavy* VSM *approximate-but-fully-learnable* is indeed a false one, and we sketched how a tiny initial linguistic structure can help bootstrapping a large-scale parsing system. We are confident through 6 years of experience, deployments and publications that RSc shops can benefit from it, and we hope *this* paper will start a discussion with participants coming from different backgrounds. While this work hardly constitutes the last word on the topic, it is hopefully a first step in leading the field away from local optima, and embracing the peculiarities and opportunities of product search.

ACKNOWLEDGMENTS

The dry prose of a scholarly paper cannot do justice to the adventure that is building an early-stage startup: this paper would not have been possible without *Tooso*, the company pioneering search-as-parsing at scale back in 2018-2019. We wish to thank first Mattia, Luca, Andrea, Alessia, and then everybody else involved in that

[14] Anecdotally, note that *ChatGPT* response to the prompt "You are a shopper assistant at Best Buy, the famous electronic retailer. You work in the video-game section. A shopper comes to you and ask for *nintendo switch*: what product do you think she wants to buy?" is "If a shopper comes asking for a Nintendo Switch, it's most likely that they are referring to the Nintendo Switch console itself".

[15] Five years after the deployment of the system in *this* paper, it is telling that leading tech retailers are starting to use a product graph for rewriting as well [19].

clumsy, special company: a challenge we were willing to accept, one we were unwilling to postpone, and one we intended to win.

Furthermore, we wish to thank Tracy Holloway King, Federico Bianchi, Patrick John Chia and two anonymous reviewers for useful comments to a previous version of this paper.

REFERENCES

[1] Qingyao Ai and Lakshmi Narayanan.R. 2021. Model-Agnostic vs. Model-Intrinsic Interpretability for Explainable Product Search. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management* (Virtual Event, Queensland, Australia) *(CIKM '21)*. Association for Computing Machinery, New York, NY, USA, 5–15. https://doi.org/10.1145/3459637.3482276

[2] Dan Alaimo. 2018. *87% of shoppers now begin product searches online.* https://www.retaildive.com/news/87-of-shoppers-now-begin-product-searches-online/530139/

[3] Dara Bahri, Yi Tay, Che Zheng, Donald Metzler, and Andrew Tomkins. 2020. Choppy: Cut Transformer for Ranked List Truncation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) *(SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 1513–1516. https://doi.org/10.1145/3397271.3401188

[4] David Basin, Yves Deville, Pierre Flener, Andreas Hamfelt, and Jørgen Nilsson. 2004. Synthesis of Programs in Computational Logic. *Program Development in Computational Logic* 3049, 30–65. https://doi.org/10.1007/978-3-540-25951-0_2

[5] David Berg, Ravi Kiran Chirravuri, Romain Cledat, Savin Goyal, Ferras Hamad, and Ville Tuulos. 2019. *Open-Sourcing Metaflow, a Human-Centric Framework for Data Science.* https://netflixtechblog.com/open-sourcing-metaflow-a-human-centric-framework-for-data-science-fa72e04a5d9

[6] Federico Bianchi, Ciro Greco, and Jacopo Tagliabue. 2021. Language in a (Search) Box: Grounding Language Learning in Real-World Human-Machine Interaction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* Association for Computational Linguistics, Online, 4409–4415. https://doi.org/10.18653/v1/2021.naacl-main.348

[7] Federico Bianchi, Jacopo Tagliabue, and Bingqing Yu. 2021. Query2Prod2Vec: Grounded Word Embeddings for eCommerce. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Papers.* Association for Computational Linguistics, Online, 154–162. https://doi.org/10.18653/v1/2021.naacl-industry.20

[8] Federico Bianchi, Jacopo Tagliabue, Bingqing Yu, Luca Bigon, and Ciro Greco. 2020. Fantastic Embeddings and How to Align Them: Zero-Shot Inference in a Multi-Shop Scenario. In *Proceedings of the SIGIR 2020 eCom workshop, July 2020, Virtual Event, published at http://ceur-ws.org (to appear).* https://arxiv.org/abs/2007.14906

[9] Bloomreach. 2022. *With $175 Million in Funding, Bloomreach Is Authoring the Next Chapter of E-Commerce.* https://www.bloomreach.com/en/blog/2022/with-usd175-million-in-funding-bloomreach-is-authoring-the-next-chapter-of-e-commerce

[10] Eliot Brenner, Jun Zhao, Aliasgar Kutiyanawala, and Zheng Yan. 2018. End-to-End Neural Ranking for eCommerce Product Search: an Application of Task Models and Textual Embeddings. *ArXiv* abs/1806.07296 (2018).

[11] Patrick Chia, Giuseppe Attanasio, Federico Bianchi, Silvia Terragni, Ana Magalhães, Diogo Goncalves, Ciro Greco, and Jacopo Tagliabue. 2022. Contrastive language and vision learning of general fashion concepts. *Scientific Reports* 12 (11 2022). https://doi.org/10.1038/s41598-022-23052-9

[12] Patrick John Chia, Jacopo Tagliabue, Federico Bianchi, Ciro Greco, and Diogo Goncalves. 2022. "Does it come in black?" CLIP-like models are zero-shot recommenders. In *Proceedings of the Fifth Workshop on e-Commerce and NLP (ECNLP 5).* Association for Computational Linguistics, Dublin, Ireland, 191–198. https://doi.org/10.18653/v1/2022.ecnlp-1.22

[13] Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K. Reddy. 2022. ANTHEM: Attentive Hyperbolic Entity Model for Product Search. In *WSDM '22: The Fifteenth ACM International Conference on Web Search and Data Mining, Phoenix, AZ, USA, February 21-25, 2022* (Phoenix, AZ, USA) *(WSDM '22)*. Association for Computing Machinery, New York, NY, USA.

[14] Big Commerce. 2021. *How Ecommerce Site Search Can Create a Competitive Advantage.* https://www.bigcommerce.com/articles/ecommerce/site-search/#the-effectiveness-of-ecommerce-site-search-

[15] Scott Compton. 2021. *Searching For ROI In Retail: The Time For A New Site Search Tool Is Now.* https://www.forrester.com/blogs/searching-for-roi-in-retail-the-time-for-a-new-site-search-tool-is-now/?categoryid=a89c0000000AKp1AAG

[16] Ethan Cramer-Flood. 2020. *Global Ecommerce 2020. Ecommerce Decelerates amid Global Retail Contraction but Remains a Bright Spot.* https://www.emarketer.com/content/global-ecommerce-2020

[17] Andrew Drozdov, Nathanael Scharli, Ekin Akyuurek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou. 2022. Compositional Semantic Parsing with Large Language Models. *ArXiv* abs/2209.15003 (2022).

[18] Mihail Eric. 2022. MLOps Is a Mess But That's to be Expected. https://www.mihaileric.com/posts/mlops-is-a-mess/

[19] Shahla Farzana, Qunzhi Zhou, and Petar Ristoski. 2023. Knowledge Graph-Enhanced Neural Query Rewriting. In *Companion Proceedings of the ACM Web Conference 2023* (Austin, TX, USA) *(WWW '23 Companion)*. Association for Computing Machinery, New York, NY, USA, 911–919. https://doi.org/10.1145/3543873.3587678

[20] Daniel Gillick, Alessandro Presta, and Gaurav Singh Tomar. 2018. End-to-End Retrieval in Continuous Space. *arXiv preprint arXiv:1811.08008* (2018).

[21] Vivek Gupta, Harish Karnick, Ashendra Bansal, and Pradhuman Jhala. 2016. Product Classification in E-Commerce using Distributional Semantics. In *COLING*.

[22] Wilfrid Hodges. 2022. Model Theory. In *The Stanford Encyclopedia of Philosophy* (Spring 2022 ed.), Edward N. Zalta (Ed.). Metaphysics Research Lab, Stanford University.

[23] Binyuan Hui, Xiang Shi, Ruiying Geng, Binhua Li, Yongbin Li, Jian Sun, and Xiaodan Zhu. 2021. Improving Text-to-SQL with Schema Dependency Learning. *ArXiv* abs/2103.04399 (2021).

[24] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Wenliang Dai, Andrea Madotto, and Pascale Fung. 2022. Survey of Hallucination in Natural Language Generation. *Comput. Surveys* 55 (2022), 1 – 38.

[25] Robin Jia and Percy Liang. 2016. Data Recombination for Neural Semantic Parsing. *ArXiv* abs/1606.03622 (2016).

[26] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Yu Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. *ArXiv* abs/2004.04906 (2020).

[27] Tracy Holloway King. 2023. *White Roses, Red Backgrounds: Bringing Structured Representations to Search.* Springer International Publishing, Cham, 191–215. https://doi.org/10.1007/978-3-031-21780-7_9

[28] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML '01)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 282–289.

[29] Brenden M. Lake and Gregory L. Murphy. 2020. Word meaning in minds and machines. *Psychological review* (2020).

[30] Rui Li, Yunjiang Jiang, Wenyun Yang, Guoyu Tang, Songlin Wang, Chaoyi Ma, Wei He, Xi Xiong, Yun Xiao, and Yihong Eric Zhao. 2019. From Semantic Retrieval to Pairwise Ranking: Applying Deep Learning in E-commerce Search. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (2019).

[31] Nelson F. Liu, Tianyi Zhang, and Percy Liang. 2023. Evaluating Verifiability in Generative Search Engines.

[32] Stefanie Marotta. 2021. *Canada's Latest Tech Public Debut Swings Amid Soft IPOs.* https://www.bloomberg.com/news/articles/2021-11-25/canada-s-latest-tech-public-debut-swings-amid-slew-of-soft-ipos

[33] Bhaskar Mitra and Nick Craswell. 2018. An Introduction to Neural Information Retrieval. *Foundations and Trends® in Information Retrieval* 13, 1 (December 2018), 1–126. https://www.microsoft.com/en-us/research/publication/introduction-neural-information-retrieval/

[34] Piero Molino and Christopher Ré. 2021. Declarative Machine Learning Systems: The Future of Machine Learning Will Depend on It Being in the Hands of the Rest of Us. *Queue* 19, 3 (jun 2021), 46–76. https://doi.org/10.1145/3475965.3479315

[35] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy J. Lin. 2019. Multi-Stage Document Ranking with BERT. *ArXiv* abs/1910.14424 (2019).

[36] Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Wenwu Ou, and Dan Pei. 2019. Personalized Context-aware Re-ranking for E-commerce Recommender Systems. *ArXiv* abs/1904.06813 (2019).

[37] Alexander J. Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid Training Data Creation with Weak Supervision. *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases* 11 3 (2017), 269–282.

[38] Borja Requena, Giovanni Cassani, Jacopo Tagliabue, Ciro Greco, and Lucas Lacasa. 2020. Shopper intent prediction from clickstream e-commerce data with minimal browsing information. *Scientific Reports* 10 (2020), 2045–2322. https://doi.org/10.1038/s41598-020-73622-y

[39] Andy Rosenbaum, Saleh Soltan, Wael Hamza, Amir Saffari, Marco Damonte, and Isabel Groves. 2022. CLASP: Few-shot cross-lingual data augmentation for semantic parsing. In *AACL-IJCNLP 2022.* https://www.amazon.science/publications/clasp-few-shot-cross-lingual-data-augmentation-for-semantic-parsing

[40] G. Salton, A. Wong, and C. S. Yang. 1975. A Vector Space Model for Automatic Indexing. *Commun. ACM* 18, 11 (nov 1975), 613–620. https://doi.org/10.1145/361219.361220

[41] Amy Schade and Jakob Nielsen. 2022. *Ecommerce User Experience Vol. 05: Search.* https://www.nngroup.com/reports/ecommerce-ux-search-including-faceted-search/

[42] Varun Shenoy. 2023. GraphGPT. https://github.com/varunshenoy/graphgpt.

[43] Daria Sorokina and Erick Cantu-Paz. 2016. Amazon Search: The Joy of Ranking Products. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Pisa, Italy) *(SIGIR '16)*. Association for Computing Machinery, New York, NY, USA, 459–460. https://doi.org/10.1145/2911451.2926725
[44] Jacopo Tagliabue. 2021. You Do Not Need a Bigger Boat: Recommendations at Reasonable Scale in a (Mostly) Serverless and Open Stack. In *Proceedings of the 15th ACM Conference on Recommender Systems* (Amsterdam, Netherlands) *(RecSys '21)*. Association for Computing Machinery, New York, NY, USA, 598–600. https://doi.org/10.1145/3460231.3474604
[45] Jacopo Tagliabue. 2022. Applied Research at Reasonable Scale. https://medium.com/the-techlife/applied-research-at-reasonable-scale-8a74d2beed89. [Online; accessed 19-Feb-2023].
[46] Jacopo Tagliabue. 2022. MLOps without Much Ops. https://towardsdatascience.com/mlops-without-much-ops-d17f502f76e8
[47] Jacopo Tagliabue, Hugo Bowne-Anderson, Ville Tuulos, Savin Goyal, Romain Cledat, and David Berg. 2023. Reasonable Scale Machine Learning with Open-Source Metaflow. arXiv:2303.11761 [cs.LG]
[48] Jacopo Tagliabue, Ciro Greco, Jean-Francis Roy, Federico Bianchi, Giovanni Cassani, Bingqing Yu, and Patrick John Chia. 2021. SIGIR 2021 E-Commerce Workshop Data Challenge. In *SIGIR eCom 2021*.
[49] Jacopo Tagliabue, Bingqing Yu, and Marie Beaulieu. 2020. How to Grow a (Product) Tree: Personalized Category Suggestions for eCommerce Type-Ahead. In *Proceedings of The 3rd Workshop on e-Commerce and NLP*. Association for Computational Linguistics, Seattle, WA, USA, 7–18. https://doi.org/10.18653/v1/2020.ecnlp-1.2
[50] Yi Tay, Vinh Q. Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. 2022. Transformer Memory as a Differentiable Search Index. arXiv:2202.06991 [cs.CL]
[51] Techcrunch. 2021. *Search API startup Algolia raises $150 million at $2.25 billion valuation*. https://techcrunch.com/2021/07/28/search-api-startup-algolia-raises-150-million-at-2-25-billion-valuation/
[52] Manos Tsagkias, Tracy Holloway King, Surya Kallumadi, Vanessa Murdock, and Maarten de Rijke. 2020. Challenges and Research Opportunities in eCommerce Search and Recommendations. In *SIGIR Forum*, Vol. 54.
[53] Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a Semantic Parser Overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, 1332–1342. https://doi.org/10.3115/v1/P15-1129
[54] Yaxuan Wang, Hanqing Lu, Yunwen Xu, Rahul Goutam, Yiwei Song, and Bing Yin. 2021. QUEEN: Neural query rewriting in e-commerce. In *The Web Conference 2021*. https://www.amazon.science/publications/queen-neural-query-rewriting-in-e-commerce
[55] Yan Yan, Zitao Liu, Meng Zhao, Wentao Guo, Weipeng P. Yan, and Yongjun Bao. 2018. A Practical Deep Online Ranking System in E-commerce Recommendation. In *ECML/PKDD*.
[56] Bingqing Yu, Jacopo Tagliabue, Ciro Greco, and Federico Bianchi. 2020. "An Image is Worth a Thousand Features": Scalable Product Representations for In-Session Type-Ahead Personalization. In *Companion Proceedings of the Web Conference 2020* (Taipei, Taiwan) *(WWW '20)*. Association for Computing Machinery, New York, NY, USA, 461–470. https://doi.org/10.1145/3366424.3386198
[57] Yongfeng Zhang and Xu Chen. 2018. Explainable Recommendation: A Survey and New Perspectives. *Found. Trends Inf. Retr.* 14 (2018), 1–101.
[58] Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint Extraction of Entities and Relations Based on a Novel Tagging Scheme. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, 1227–1236. https://doi.org/10.18653/v1/P17-1113

A IMPLEMENTATION NOTES

As the novelty of our proposal does not lie in new classifiers or NLP pipelines, we briefly expand here the implementation strategies sketched in Section 4. We count as a strength of the approach that tried-and-tested and off-the-shelf techniques can be successfully used to start: any improvements to the below methods will make the parser even better.

A.1 Product representation

Once basic tags (COLOR, BRAND, etc.) are defined as the building blocks of the knowledge base *and* the logical forms, we need to know where and how each of these attributes can be found starting from the product catalogs. We favor a declarative approach, where tags are associated with strategies, that get executed in series when parsing the catalog. For example, Table 1 shows how three tags can be extracted from Shop A.

Tag	Type	Strategy
Brand	Config	*Manufacturer*
Color	Model	CLIP-based classifier
Product	Heuristic	First noun in *Description* overlapping with *Category*

Table 1: Building a knowledge base for Shop A. Three sample strategies for building symbolic product representations using naming conventions, machine learning, and domain specific heuristics.

We first have *configuration* strategy, which just points to the column in the catalog that contains the attribute (typical for brands, prices etc.); this leverages the structured nature of catalogs, which is a huge simplifying factor when considering product search *vis-à-vis* web search. We then have a *model* strategy, which relies on machine learning to accomplish tagging; finally we have a *heuristic* strategy, building on domain knowledge and catalog specifics.

When discussing scaling B2B product search across deployments, it's important to realize different strategies have different levels of granularity. Configurations are set *per shop* and they are deterministic; models can typically be trained across shops (for entire industries for example) and can leverage the latest zero-shot classifiers in case no label is wanted / needed [11] *heuristics* are more case specifics, but in our experience they have some degree of re-use: moreover, heuristics can be used to train new classifiers (using for example weak supervision [37]), which will in turn reduce the use of heuristics.

Importantly, the very recent progress on large language models promises to greatly simplify the actual building of a structured knowledge representation, offering even zero-shot graph building from text [42]. While LLMs are still too slow and somehow not understood enough to be directly involved in the runtime query path, they are definitely well suited to speed up the offline component of our method (Fig. 1, section 1 and 2 from the left).