Managing exposure of Web content to AI systems

by Dominique Hazaël-Massieux <dom@w3.org>, July 26 2024, Position Paper for the IAB Workshop on AI-CONTROL

As we describe in our report AI & the Web: Understanding and managing the impact of Machine Learning models on the Web, the scale of processes involved in the building and deployment of recent large Machine Learning (ML) models is such that they are now set to have a *systemic* impact on the Web as a shared information space.

Among these high-scale processes, the usage of content and data crawled from the public Web to train very large ML models is disrupting several (often already precarious) balances between content producers and large scale content-access intermediaries such as search engines, social networks, content aggregators, including the intermediation newly provided by AI chat interfaces.

`robots.txt` has so far been one of the primary entry points to managing that existing balance, whereby a content producer could indicate which part of their content could be crawled and by which crawler(s) (as identified by their `User-Agent` header).

While that makes it a critical part of the solution that needs to emerge to find a new sustainable balancing point, its design has several structural limitations that makes it challenging in fulfilling that role.

`robots.txt` was initially designed to manage the load induced by crawlers on the server hosting the content. Because the primary and most impactful usage of these crawlers was linked to indexing by search engines, `robots.txt` has primarily evolved to help coordination between content producers and search engines crawlers with server load mostly solved through other mechanisms.

There are clearly commonalities between the scale of crawlers search engines and AI systems need. These commonalities are, somewhat confusingly, strengthened by the increasing conflation of the two kind of systems from a user perspective: search engines increasingly expose ML outputs in addition to search results, and users increasingly use queries to AI systems (e.g., via chat interfaces) to accomplish information retrieval tasks for which they would previously have used a search engine.

`robots.txt` is also well-understood by the ecosystem, and has already been leveraged to manage (and particularly disallow, e.g., 28% of most actively maintained source of the C4 crawl) crawlers used to train AI systems.

Nevertheless, there are also currently stark differences between the underlying systems that limits its fitness to the issue at hand.

Reversibility of consent on crawling

The crawl data used by search engines is typically kept private, centralized and under full control of the search engine operator. In particular, these centralized datasets can be (and are regularly)

curated by these operators to reflect changes in the underlying content policy or respond to legal requests (incl. privacy and copyright infringements).

In comparison, a number of AI systems are trained in significant part over crawl datasets that are publicly available (e.g., Common Crawl and derivatives), with no single point of control once that data is distributed.

The ML models trained over these datasets can be reasonably considered in first approximation as lossy compressions of the crawled datasets. Given that these models are themselves distributed and exploited in a wide variety of ways, there is very limited control over content and data that have found its way in trained models. And since training a model is mostly a non-reversible process, it is essentially impossible to have one's content robustly removed from a model.

This **lack of reversibility** raises the stakes of opening up one's content to crawlers that *can* be used to train ML models, in comparison to search engines.

The opt-out nature of `robots.txt` is not a great fit in such a situation. The focus on identifying crawlers rather than the systems that get fed with crawled content further down the line ends up with mismatches:

- crawlers whose data have been and are being used for other purposes get banned to prevent that particular re-use;
- and with many new players coming in the market to satisfy the need of ML training, the proliferation of crawlers user agents make it hard to robustly use `robots.txt` to prevent that particular re-use.

Quid pro quo of crawler access

While the relationship between search engines and content providers has never been an easy one, the quid pro quo of search engines providing access and visibility to these providers' content in exchange of making their content indexable has proved sufficient to sustain a rich ecosystem of content producers, even if there are legitimate critiques of the power imbalance that arises from the critical role of intermediary played search engines.

The intermediation role played by AI systems is altogether new: where the role of search engines has traditionally been to surface the most relevant links to answers of the user's query, AI systems typically expose *directly* an answer; that answer may or may not link to sources that support it, and even when it does, it is bound to provide a much smaller set of references, and given the lossy nature of training, one that is likely much less tractable by content providers.

For the large number of content producers whose sustainability relies on direct exposure to (or interactions with) the final end user, this **lack of reliable exposure** makes it unappealing to leave their content crawlable for AI-training purposes.

`robots.txt` was built primarily to serve a quid pro quo based on indexability; the binary control it surfaces (`Allow` and `Disallow`) reflects the relative simplicity of that expectation. It is not obvious that the much more challenging imbalance between AI systems and content producers can be solved by retro-fitting it with new features.

Beyond crawlers

For the most part, a search engine reflects the static state of previously crawled content, with no interaction with the said content at the time of the user query.

A number of emerging AI systems (e.g., chat bots) exploit Web content in their two main operational phases: statically during their training phase, based on the crawled content used to feed it; and dynamically during their inference phase, where the live content of a page can be used as direct input (e.g., “summarize this page”).

From a `robots.txt` angle, these two phases of content usage make use of two very different type of agents: in-scope *crawlers* (automated clients) vs out-of-scope *fetchers* (client triggered by a user request - to re-use the terminology used to categorize Google's non-browser user agent strings.)

From a content provider perspective, AI systems provide a thick layer of intermediation in both phases: they are likely to want to manage access to their content from a given AI system from a unified configuration. This mismatch has been recently illustrated by by the recent outcry on a chatbot not respecting `robots.txt`, where arguably the user-triggered request didn't match the scope of `robots.txt`.

The **limitation of focusing on crawlers** as the primary control point of AI systems is made even starker by other emerging changes: not only have search engines started exposing AI systems in their UIs, but browsers are doing the same, both directly in their own user interface, but also exposing APIs to run ML models (e.g., Chrome's experimental Prompt API) - these APIs could thus be run on any content fetchable from any Web site - at which point a content producer wanting to control the use of their content in inferencing tasks would need to integrate the Cross-Origin Resource Sharing protocol in their considerations.

Integrating `robots.txt` into a usage consent framework

`robots.txt` evolved in the very early days of the Web to respond first and foremost to technical constraints. That role has evolved over time driven by the need to manage the (often complicated) relationship between content producers and search engines.

Its rapid adoption to disallow crawlers identified as feeding AI systems illustrate both the need for content producers for a solution in that space and the fact that `robots.txt` can play at least a stop-gap role in letting content producers express their intent towards that new usage.

Additions to `robots.txt` rules that would both **make it more robust to new entrants in the AI crawler spaces** and **more nuanced on what is disallowed** to generic-purpose crawlers would provide a useful breathing space to develop a more comprehensive framework.

Such a framework would usefully enable content producers to express more nuances in the level of comfort they have in making their content available to Web agents in relation to the distance of direct end user interaction.

Some of that framework would like need to be surfaced into client-side technologies - some of the semantics of `robots.txt` are already exposed and refined in HTML via `<meta name=robots>` declarations; if models gain further traction in their integration as primitives of the client-side platform, ensuring they do so in a way that can respect this broader framework will be critical.