Semantic Web: Learning from Machine Learning

Dan Brickley

Google

Abstract. Recent breakthroughs in the machine learning community have sometimes been characterized as a kind of steamroller. Having first crushed earlier results in image understanding, deep learning researchers have also been making impressive progress with both audio and video understanding and generation, and have had high profile and evocative results with natural language processing. While it is not surprising if this trend can seem threatening (in the examples given, it threatens to overturn previous methods) to a number of traditional subfields of AI (and their associated approaches), Semantic Web researchers and advocates would be ill-advised to consider deep learning a rival or threatening effort. In fact, we argue that there is much to learn from the success of these methods.

Introduction

This brief (opinionated) position paper offers a few observations on Machine Learning, and in particular recent developments around Deep Learning. It is addressed primarily to implementers, researchers and technologists working on Semantic Web and related structured data systems.

The Semantic Web initiative, above all, was a project whose goal was to help improve the Web, and to provide a common general-purpose explicit structure for Web content that can remove some of the guesswork from handling it. In no way is this set up in opposition to deep learning and other machine learning approaches. In a research setting, especially considered as competing paradigms for understanding natural intelligence, or for simulating it, it is common to see explicit, logically-grounded declarative knowledge represention contrasted with its presumed ”rivals” that emphasise numerical, vector, statistical or other continuous representations such as the weights and activations of an artificial neural network. We now discuss how these initiatives are in many ways complementary. We then discuss how the Semantic Web project can learn from the the machine learning community.

Goals

When considered as a practical Web technology, much of this apparent rivalry dissolves. The goal of RDF (and RDFS, OWL etc.) is not to create an artificial intelligence, nor an improved understanding of the human mind, but to make data sharing with the Web easier and more robust. This does not mean that people building things with RDF have nothing to learn from the cognitive sciences, of course. To the extent that RDF-based techniques are useful as data-handling technologies, they ought to be entirely complementary to the recent breakthroughs achieved via deep learning. Semantic Web can provide formats for data and metadata that is processed by ML systems (just as it can do the same for systems that are built from Perl, XSLT, SQL or that even have humans in the loop). It can also provide formats for the representation of the outputs of such systems - either as simple crude factual assertions (the classic use of triples) or through various forms of reification, qualification and annotation.

Follow the datasets

It is noteworthy that many of the well publicized breakthroughs in machine learning, from the recent past have happened in large part because of the availability of large datasets (together with the computing resources to process these datasets) versus the development of completely new algorithms.

Modern machine learning (ML) systems are only as good as the data they learn from. It is all too easy to grab any tabular dataset, load it up into a Jupyter Notebook and feed it to one of the numerous and powerful ML libraries that are out there. Often this will achieve very interesting and useful results, but there has been growing appreciation of the risk of ML systems reproducing and propagating into applications a variety of biases and common failings that were present in the source data and which were not properly noticed or understood by the ML application developer. Often data journalists, fact checkers, social scientists and other domain experts will have a more nuanced understanding of the strengths and weaknesses of a dataset than an ML specialist who is new to some domain. Semantic Web formats and approaches (e.g. schema.org and W3C DCAT dataset descriptions, CSVW, Data Cube etc.) provide mechanisms that can be used to annotate columns, cells and entire datasets with explicit footnotes or caveats indicating different kinds of concern (eg. measurement accuracy, related data-distorting real world events, provenance) that can help provide more real world background knowledge to ML workflows.

Probability and triples

The machine learning community was one of the first to employ representations that incorporated probabilities. It is clear that we need to find ways of representing probabilities in the Semantic Web community as well. Probabilistic models for RDF face the challenge that real world triples are too small and ambiguous a data structure to use for probabilities. Ascribing probabilities to entire RDF graphs is also problematic, since they are typically too big and messy.

RDF Shape expression languages (e.g. W3C SHACL, Shex) may turn out to be a good fit for this problem. By being larger than a triple they allow the meaning of blank nodes to be fixed through the semantics of neighboring triples, rather than left as wild cards. By being more regular and disciplined than giant messy graphs, it is more likely that well have enough occurrences of the same graph pattern to reason over interestingly.

The problem with probability and triples is that unless every term in the triple is grounded, then the entity being referred to is ambiguous or unknown since the content of other properties are required to understand its meaning. Consider the difference between representing the probability of ”bnode-id-12616135241 type Event” versus (a different but related claim) ”An entity of type MusicGroup whose name is ’Madness’ has a MusicEvent event on 2010-05-15” in a Place whose name is ’Amsterdam’”. By using a graph structure larger than a triple, the semantics of the propoposition whose probability is being represented are much clearer.

Beyond triples

Semantic Web representations have their roots in First Order Predicate Calculus based knowledge representation formalisms. At best, there were a few thousand practitioners of these comparatively complex knowledge representation formalisms. With the advent of the web, there was a pressing need for something more comprehensible by much larger constituency. The goal was to adopt the declarative semantics of these systems, without necessarily bringing along the whole inference baggage. Triples were a great first simplification that could be explained (and thus adopted) by millions.

But as any implementor of these triple systems knows, one has to go beyond triples, sooner rather than later. There is the need to accommodate time, provenance and possibly other annotations or extensions of triples. There is often a need to be able to compose concepts. E.g., given a numeric attribute (say height), one might want max(height), average(height) and so on. These and other extensions which are usually implemented, often with a bit of guilt, under the wraps. The problem with this is that these extensions are now beyond the scope of what can be exchanged between these systems.

Note that each of Schema.org, Freebase, Wikidata amongst others have all wrestled with the expressiveness problems associated with triples, and have each created similar but different idioms for putting additional information within the triples model.

Triples have the convenient property that you always know their arity; consider a simple RDF property like worksWith or livesWith. In RDF, livesWith is always livesWith/2; there is nowhere to put extra information, so we don’t need to worry how that extra information relates to simplest form of the predicate. If we are to see livesWith/4 /5 or /3, a post-triples design would need to be clear on whether additional information is always additive, and whether livesWith of other arities could be concluded from livesWith/4, for example.

Whats in a class?

Languages such as OWL have emphasized tightly-defined logical definitions for classes, sometimes going so far as listing necessary and sufficient conditions for class membership, expressed in terms of the properties of some candidate entity. For some applications this can be very useful. However, for applications whose scope extends across the entire Web, it is often more pragmatic to leave some flexibility in the definitions. When defining a class such as Event, or Person, its semantics are often more usefully grounded through real world applications, and through natural language, rather than in logic-based definitions. Notwithstanding the comments made above about the neural networks vs formal semantics pseudo-rivalry, the recent successes around deep learning may add an additional argument here. If human ability to use and understand complex and subtle concepts such as ”person”, ”event”, ”action”, ”restricted diet” or ”government permit” without having direct explicitly articulated rules is in fact grounded into something along the lines of a neural net ”weights and activations” representation, perhaps we need to get used to the idea that such concepts in their human use aren’t based on anything like explicit definitions. And so, when defining RDF classes for humans to use to build human-facing applications, perhaps we can relax our expectations correspondingly.

Embeddings and model sharing

Over the last couple of years, various approaches based on the idea of embeddings have evolved, in order to represent words, concepts, relations, etc. We have seen some widely used embeddings, such as Glove, that are trained on very large corpora. At the same time, we have seen the widespread adoption of more ‘Semantic Web’ standards such as Schema.org. It would be interesting if we could relate the two.

One aspect to explore would be to consider trained ML models as datasets for publication, exchange, re-use and combination in the Web. It is already common practice in ML workflows to start from a basic pre-trained model created on a related dataset (e.g. imagenet for photos) and then do further training (e.g. of additional neural network layers) for some more focussed task. Currently there are no strong descriptive conventions or data-sharing habits to support this. Typically a pre-trained model is available online via a ”Zoo” page in a Wiki or on Github, and the published serialized representation is a collection of files tightly bound to a particular release (and configuration e.g. GPU vs CPU) of a particular machine learning toolkit such as Tensorflow or Torch.

Even documenting more clearly these kinds of datasets and their dependencies could be a very useful contribution to the research environment. Shared ML models have some of the characteristics of software, and some of data. At this time it seems unclear whether they are likely to be shared more using the conventions of software publication and distribution (e.g. github, debian, ruby gems, Docker etc.) or of data (repositories and portals, data citation conventions). Deep learning ML models are significantly harder to inspect or understand than traditional software artifacts. This has serious security considerations (as well as legal e.g. copyright and the notion of a derived work). For example, consider a pre-trained model that classifies a typical Imagenet task, but has been adapted to recognize images of the present author and classify them as President. Such a network could still be useful at its original task and could prove popular and widely used. Verifying and avoiding such unwanted and possibly malicious hidden functionality within ML models is a relatively new field. It may be that supporting metadata e.g. tying checksums to provenance and sourcing / certification information may help mitigate some concerns here.

Many mappings

For some reason, RDF and Semantic Web technologies seem particularly appealing to generalists. While this underpins some of the strengths of the approach, it can also sometimes be damaging, i.e. by fostering a culture in which pragmatic, case-by-case solutions can be dismissed as unprincipled or as conflicting with some imagined Semantic Web architectural design principle. It is therefore worth stating very explicitly that we should expect there to be a very wide variety of mappings from ‘Semantic Web data’ (whatever exactly that is) into deep learning (i.e. vector based) models, and that such variety is a healthy sign. A strength of the current deep learning research community is the variety of representational styles, architectural approaches and learning algorithms being explored. We should expect and encourage similar pluralism with any work that maps complex data from RDF triples into somehow more regular vector-based form. It may be that over time an informal collection of reusable patterns emerge, but it would be unfortunate if these became dogmatically imposed during a time of rapid innovation, experimentation and discovery.

For example, many highly regular datasets that are represented in RDF, are so 'vanilla' or mainstream in their structure that their 'RDF-ness' becomes unremarkable and somewhat irrelevant. A simple and flat records-and-fields dataset might be encoded in RDF, in plain CSV, or custom XML/JSON. Even its RDF representations could be in terms of entities and their properties (e.g. houses for sale) or in some reified structure such as DataCube or CSVW.

RDF datasets in some sense are always graphs. However there is another sense in which RDF might carry graph data, such as when the content represents or describes a real world network structure. For example in FOAF (and also schema.org) RDF organized around Person nodes linked by relations such as ”knows”, ”follows”, ”funder” etc. form a core data structure. There is already a huge literature on social network data mining, and on the science and statistics of network data structures. While RDF is well suited as an encoding of such data, it is as ever one of several ways that social network and similar information can be shared. A popular representation for such data (and more broadly other graphs with weighted connections) is the Laplacian graph matrix, alongside related techniques from spectral graph theory and spectral clustering. This is just one example of a case where domain specific representational structure can be usefully interposed between RDF source data and generic ML processing.

A variation on this data pattern is the extremely common case of RDF data representing a bipartite graph. For example, movie ratings where weighted associations link movies and users; or book/paper authorship. Or the topics of a set of events. It may be that much of the Web of Linked Data can be usefully seen as a set of superimposed bipartite graphs, and that this data structure may merit special attention when mapping into the simpler, flatter ML representations.

Time series datasets and the details of recurrent neural network systems, give another example whose complexities cannot even be fairly summarized here.

The point being made here is simply that RDF data is more or less simply data, and that it needs to be considered case-by-case rather than seeking an entirely generic workflow. Having said that, in practice RDF datasets do fall into similar clusters or patterns as indicated here, and it seems reasonable to expect community best practices to emerge that make it easier to build upon previous similar approaches.

Conclusion

There is a rich history of dialogue around the relationship between propositional and non-propositional knowledge representation going back decades (or even centuries), most recently with much of it inspired by (or reacting to hype from) ongoing innovation around artificial neural networks. The re-branding of modern artificial neural network engineering as ”deep learning” makes practical sense, but readers should be aware that there were substantial debates throughout the 80s and 90s that may be worth reading. In particular the terms ”connectionism” and ”connectionist” were widely used in that era to characterize artificial neural network models when discussed in a broader cognitive science (e.g. psychology or linguistics) context.

For a classic example of easy-overlooked history, the ”Letter to Nature” from Rumelhart et al of 1986 which helped make the back-propagation algorithm widely known featured a running sidebar example based on the representation of family tree data using triples (104 triples; this was 1986).

To conclude, a quote from 1945 [1], which remains relevant today as we explore how to move from classical document search systems towards those which can more intelligently engage with us and assist us with our lives:

‘Philosophers have not done justice to the distinction which is quite familiar to all of us between knowing that something is the case and knowing how to do things. In their theories of knowledge they concentrate on the discovery of truths or facts, and they either ignore the discovery of ways and methods of doing things or else they try to reduce it to the discovery of facts. They assume that intelligence equates with the contemplation of propositions and is exhausted in this contemplation.’

References

1. Gilbert Ryle. Knowing how and knowing that: The presidential address. In *Proceedings of the Aristotelian society*, volume 46, pages 1–16. JSTOR, 1945.