Introduction to the Policy Difference Engine - Part 2

Regan Meloche - Nov 2020

In a previous article, I introduced the project that one of the Code for Canada team's will be working on for a 10 month fellowship. TLDR: We will be working with the a government agency (ESDC) to build a policy difference engine, the high-level goal of which is to measure the impact of changing rules. The two subgoals of this project are:

- Finding relevant connections between different rules
- Running valid simulations on proposed rule changes

We also introduced the concept of Rules as Code, which involves translating written rules into something that can be consumed by a machine. This involves removing ambiguities from a rule, and splitting it into atomic logical statements. The example we highlighted was eligibility for a child disability benefit in New Zealand.

Code For Canada is a non-profit organization that works with government partners to deliver better digital services to the public. This year one of the Code For Canada teams will be working with the ESDC, which handles many public services such as employment insurance, pensions, and disability benefits. These services are bound by a wide variety of legislations, regulations, and policies, which we will generalize here under the term "rules", which often need to change. Examples may include changing an age requirement for eligibility of an allowance, changing the dollar amount of a benefit, re-defining what is meant by a "vehicle", etc. With technologies that directly impact the public growing in complexity, and the population experiencing increasingly diverse living situations, it is important to keep the rules up to date. One challenge that quickly emerges is effectively measuring the impact of changing a given rule. This is the challenge that the ESDC and Code for Canada will be eagerly addressing. At the time of writing (Nov 2020), we are at the beginning of the fellowship, which will last until August 2021. Code for Canada's goal is to work closely with the ESDC for 9 months to make measurable progress on the challenge, and then smoothly hand off the project to the ESDC at the end of the fellowship. This article is a brief intro to some of the more technical aspects of the project. We will begin by further clarifying the problem, then we will describe a general tech-agnostic solution, followed by a deeper dive into some more specific technologies that may be applicable to our solution.

The problem

The overarching challenge is to measure the impact of changing a rule. This problem statement is quite general, and can be broken down into different components, and we will focus on two of these. First of all, many rules will be inextricably linked to other rules. When a change is proposed to policy A, and policy A is linked to policy B, C, D, those three policies must also be investigated to verify if the change will have any unintended side-effects. This may further cascade to even more policies, exponentially increasing the amount of investigation. Much of this investigation is manual, so it can be difficult to keep track of all potential side-effects.

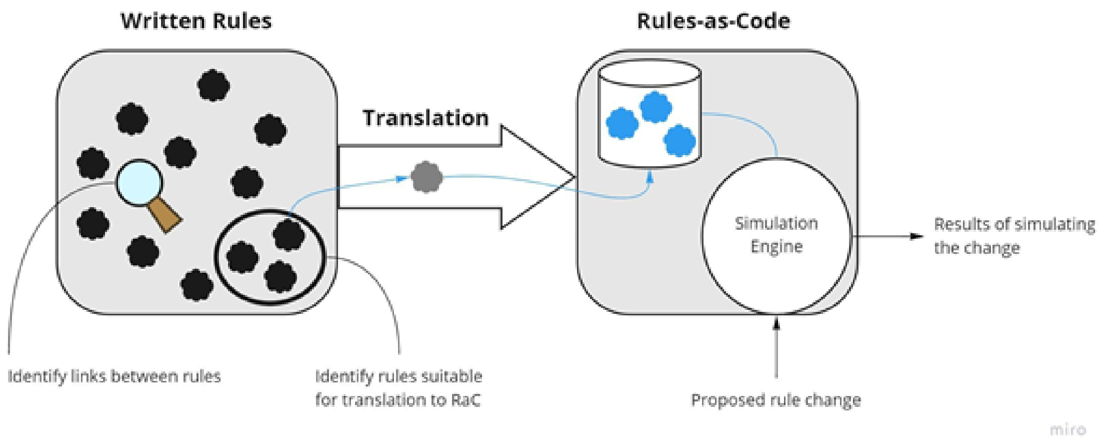
The other issue is concretely measuring the impacts that a change may have. This includes the impact on the ability for the ESDC to effectively deliver services, the impact on the cost of delivering the service, and perhaps most importantly - the impact on the people that the rule is written for. Let’s use an income tax rate as a straightforward example. How does changing an income tax rate by a certain amount affect a family of four with a monthly income of $5000? How does it affect a single person with a monthly income of < $1000. There are many cases of households and individuals that must be considered for any proposed change. If we could run simulations on a model population that approximates the population of the area in question, then we’ve made progress towards evidence-based policy. The example of changing income tax is fairly straightforward to reason about, since it largely deals with numbers. This is not the case for all legislation. How would we measure the impact of changing some policy surrounding something more complex, such as changing the definition of a “vehicle” in some rules to include the latest and trendiest means of individual transport in a big city (e.g. motorized scooters)? On the surface, this does not appear to lend itself to a simulation in the same way that a number-heavy rule such as income tax might, and this will likely prove to be one of the more challenging aspects of the project.

A Policy Difference Engine

The high level goal is to measure the impact of changing rules. The two subgoals are:

- Finding relevant connections between different rules
- Running valid simulations on proposed rule changes

With these in mind, we can design a high-level, technology-agnostic workflow of what a solution might look like, which we will refer to as a **Policy Difference Engine (PDE)**. This is a proposed tool or set of tools that may be used primarily by someone responsible for proposing changes to existing legislation, regulations, or policies, which we will refer to as a ‘rule writer’.


Written Rules
Rules-as-Code
Translation
Simulation Engine
Results of simulating the change
Identify links between rules
Identify rules suitable for translation to RaC
Proposed rule change
miro

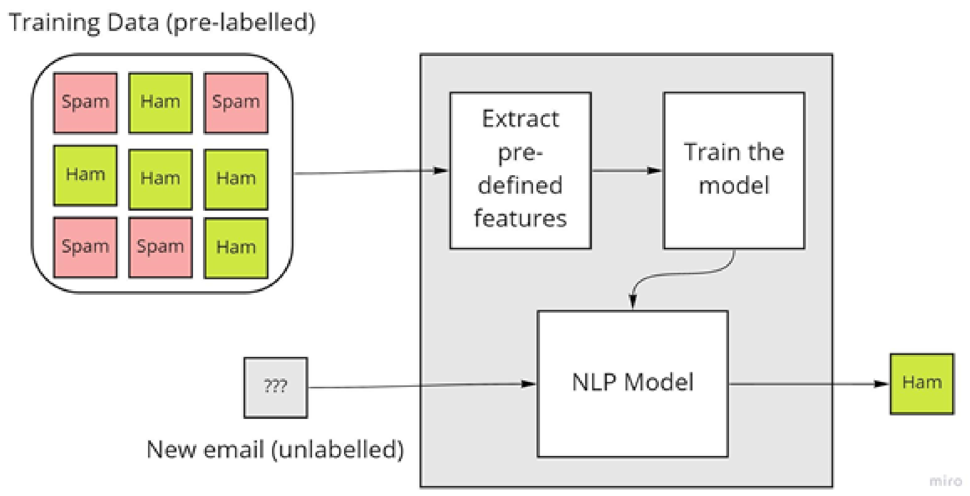
On one side we have the world of written rules. These rules in their current form cannot be assumed to be consumable by a machine for running simulations. However, it may still be possible to satisfy part of the first subgoal by tagging some rules as being related to other rules. A rule writer could use this tool when considering updating a rule to help guide them on what other rules need to be taken into consideration before even beginning to make concrete changes to the rule.

In order to satisfy the second subgoal, we must convert the written rules to RaC. Then we need to create personas in the system, toggle values in the rules to represent changes, and generate the simulation results. The form that these results take may vary depending on the rule we are working with. We may get a list of household types that are negatively impacted, a list of those positively impacted, an overall estimate of money saved/lost per household, etc. This is data that can be taken into account by rule writers to help guide the decision-making process.

At this point, we've only described a very general process. There are however, some existing technologies that present themselves as viable components for a concrete solution based on some relevant existing work.

NLP

Natural Language Processing (NLP) is the field of building systems that can understand and produce human language. Use cases include spam-filtering, chatbots, autocomplete, and sentiment analysis on product reviews and social media. The field has roots that go all the way back to the 60s (https://en.wikipedia.org/wiki/ELIZA), but it's only fairly recently that the field has taken off, due largely to the fact that we have lots of data to work with (a prerequisite for many machine learning applications) from the internet as well as faster computer speeds.



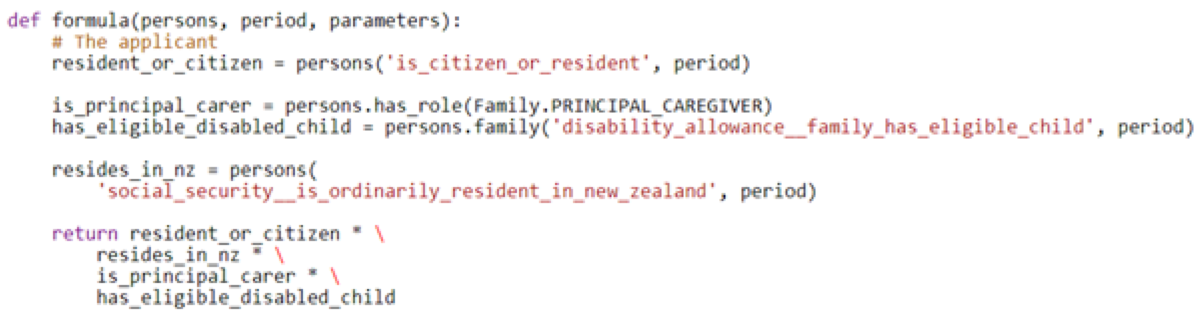
We can use the classic spam-filtering application as an example. We begin with thousands of existing emails that have been labelled as either spam or ham (not spam). This is called the 'training data'. We give this training data to an NLP system (sometimes called a pipeline) and have it build a model based on this data about what constitutes ham and what constitutes spam. We need to tell the pipeline specifically how it should process and interpret the data. We can tell it to split an email into words and sentences, and then look for words that mostly occur on spam emails and those that occur on ham emails. For example, we may find that spam emails tend to have less words per sentence than ham emails and that words like 'Prescription' and 'Health' are more likely to appear in spam than ham. So using these and other features, the pipeline builds a model based on all of the training data. Once that model is built, we can give it new examples of unlabelled emails, and using the features programmed into it from the training data, it will make a determination with a certain degree of confidence about whether the email is ham or spam.

For our purposes, NLP may be a candidate for achieving our first subgoal of identifying links between pieces of legislation. There is a case study done for the Accessibility for Ontarians with Disabilities act that uses NLP methods on legislation to achieve similar results. This study found that a serious weakness of using NLP on legislation is the lack of pre-labelled data to work with. In general, the more data that a machine learning model is trained with, the more accurate it will be. So while the field of NLP is well-established with many conventions and performant open-source libraries, this particular application of it is not. This has the potential to be a technical roadblock as we build out a solution to the PDE.

An essential piece of the general solution is the translation from written rules to RaC. It may be tempting to think that NLP could be used to do this conversion, but in reality this will likely be a very difficult task. The actual translation will be largely manual, but NLP may be able to provide some degree of assistance in the task. This could include helping identify pieces of written legislation that would be suitable candidates for conversion to RaC. This process might consist of looking for heavy use of numbers and equations in the written legislation. The AODA study sought to clarify the party responsible for delivering various services in legislation, so they used it to identify legislation that had some sort of obligation-related wording (words like “should”, “is obligated”, etc.).

OpenFisca

Another tool we will explore is OpenFisca, which is an open-source engine that allows us to work with RaC. It has been explored by several other countries for various applications, such as simulating the effects of proposed legislation and verifying eligibility for a certain benefit. This aligns well with the goals of our PDE, so we are hoping to build on previous work. It is worth emphasizing that this is NOT a software for converting written rules to RaC. It assumes you are using RaC to begin with, and then runs tests and simulations on them.

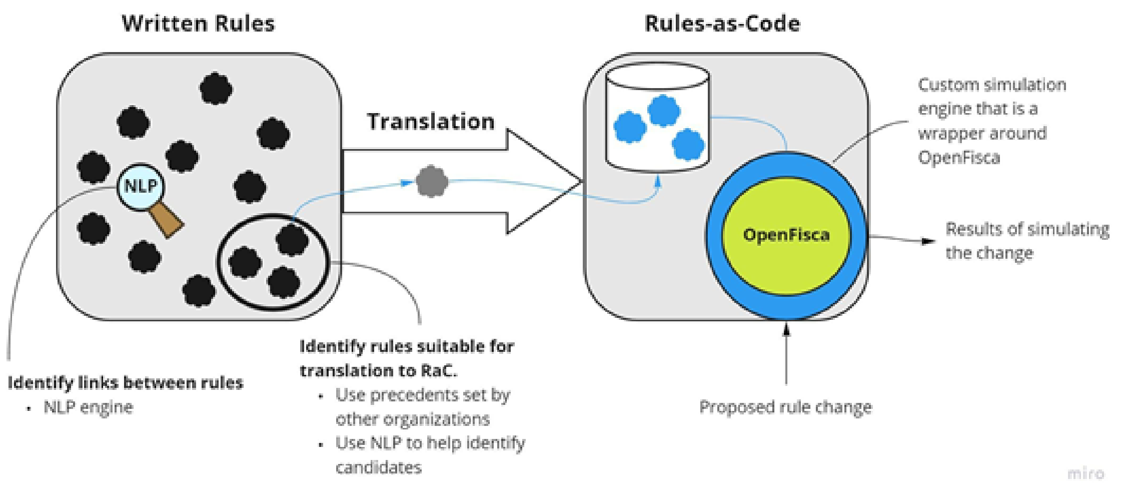


```python
def formula(persons, period, parameters):
    # The applicant
    resident_or_citizen = persons('is_citizen_or_resident', period)

    is_principal_carer = persons.has_role(Family.PRINCIPAL_CAREGIVER)
    has_eligible_disabled_child = persons.family('disability_allowance__family_has_eligible_child', period)

    resides_in_nz = persons(
        'social_security__is_ordinarily_resident_in_new_zealand', period)

    return resident_or_citizen * \
        resides_in_nz * \
        is_principal_carer * \
        has_eligible_disabled_child
```

Above is a quick sample of the code (written in Python) that calculates the eligibility for a child disability allowance that we showed earlier from the New Zealand example. This is RaC in action. It simply needs to accept the parameters based on the atomic pieces of the legislation and then we get a result.

Concrete Diagram

So now that we have some idea of what these technologies may be able to accomplish, we can build out a more concrete map of a solution as follows.



Written Rules
Rules-as-Code
Translation
NLP
OpenFisca
Custom simulation engine that is a wrapper around OpenFisca
Results of simulating the change
Proposed rule change
Identify links between rules
• NLP engine
Identify rules suitable for translation to RaC.
• Use precedents set by other organizations
• Use NLP to help identify candidates
miro

For the first subgoal of linking relevant rules, we may be able to build an NLP pipeline that can read through the rules and establish valuable links to other rules that may not have been known before. So when a rule writer wants to propose changes to policy, they can get a more holistic view of the impact that any change might have, and potentially save a significant amount of time on reading through unnecessary pieces of legislation by focusing on the closely related legislation. Note that this subgoal can be separated from the idea of RaC. There is no requirement that the legislation must be converted to a RaC format, although having that encoding may improve the performance of the engine.

The second subgoal of simulating the impact of a change requires the translation to RaC. As mentioned, this will largely be a manual process, though a separate NLP system may help identify potential candidates. We can also build on work done by other organizations in this space and target rules that have worked elsewhere. For example, France has done lots of work on tax-related rules, and we saw the New Zealand example with an eligibility-related rule. Once we've identified viable candidates for the conversion, the conversion itself will need to be done collaboratively with subject matter experts to ensure that we are preserving the legal intent of the rule. Once the legislation has been converted to RaC, we can use them inside a tool that makes use of OpenFisca to run simulations. These results can then be used by the rule writers to measure the impact of a policy change.

Next Steps

At this point we have defined the scope of the problem, proposed a general solution, and narrowed in on a few key technologies to explore further. The next steps will include performing more technical research to find more case studies and use cases, identifying key stakeholders so we can collect user input for the design, and building out proof-of-concepts for the NLP and OpenFisca components.