In this article, we present an example of an (im-)practical application of the Hidden Markov Model (HMM). It is an artificially constructed problem, where we create a case for a model, rather than applying a model to a particular case? although, maybe a bit of both.

Here, we will rely on the code we developed earlier , and discussed in the earlier article: ‘Hidden Markov Model - Implementation from scratch’, including the mathematical notation. Feel free to take a look. The story we are about to tell contains modeling of the problem, uncovering the hidden sequence and training of the model.

Tryton is a business software platform which comes with a set of modules that can be activated to make an ERP, MRP, CRM and other useful applications for organizations of any kind.
During this month, when most developers are social distancing, we recorded a lot of changes to prepare for the upcoming release 5.6 that is planned for the start of May.

The MS Word utility from Microsoft Office suite is one of the most commonly used tools for writing text documents, both simple and complex. Though humans can easily read and write MS Word documents, assuming you have the Office software installed, often times you need to read text from Word documents within another application.
For instance, if you are developing a natural language processing application in Python that takes MS Word files as input, you will need to read MS Word files in Python before you can process the text. Similarly, often times you need to write text to MS Word documents as output, which could be a dynamically generated report to download, for example.

- **Linked Lists in Python: An Introduction** [5]

  Linked lists are like a lesser-known cousin of lists. They're not as popular or as cool, and you might not even remember them from your algorithms class. But in the right context, they can really shine.

- **Python Software Foundation: An Update on PyPI Funded Work** [6]

  Originally announced at the end of 2018, a gift from Facebook Research is funding improvements for the security PyPI and its users.

- **Django bugfix releases issued: 3.0.5 and 2.2.12** [7]

  Today we've issued 3.0.5 and 2.2.12 bugfix releases.

- **Concurrency in Python** [8]

  A thread is an independent sequence of execution, but it shares memory with all the other threads belonging to your program. A Python program has, by default, one main thread. You can create more of them and let Python switch between them. This switching happens so fast that it appears like they are running side by side at the same time.

- **What the heck is pyproject.toml?** [9]

  Recently on Twitter there was a maintainer of a Python project who had a couple of bugs filed against their project due to builds failing (this particular project doesn't provide wheels, only sdists). Eventually it came out that the project was using a pyproject.toml file because that's how you configure Black and not for any other purpose. This isn't the first time I have seen setuptools users use pyproject.toml because they were "told to by <insert name of tool>"
without knowing the entire point behind the file. And so I decided to write this blog post to try
and explain to setuptools users why pyproject.toml exists and what it does as it's the future of
packaging in the Python ecosystem (if you are not a conda user).

[...]

With PEP 518 in place, tools knew what needed to be available in order to build a project into
a wheel (or sdist). But how do you produce a wheel or sdist from a project that has a
pyproject.toml? This is where PEP 517 comes in. That PEP specifies how build tools are to be
executed to build both sdists and wheels. So PEP 518 gets the build tools installed and PEP
517 gets them executed. This opens the door to using other tools by standardizing how to run
build tools. Before, there was no standardized way to build a wheel or sdist except with python
setup.py sdist bdist_wheel which isn't really flexible; there's no way for the tool running the
build to pass in environment details as appropriate, for instance. PEP 517 helped solve that
problem.

One other change that PEP 517 & 518 has led to is build isolation. Now that projects can
specify arbitrary build tools, tools like pip have to build projects in virtual environments to
make sure each project's build tools don't conflict with another project's build tool needs. This
also helps with reproducible builds by making sure your build tools are consistent.

Unfortunately this frustrates some setuptools users when they didn't realize a setup.py files
and/or build environment have become structured in such a way that they can't be built in
isolation. For instance, one user was doing their builds offline and didn't have setuptools and
'wheel' cached in their wheelhouse, so when pip tried to build a project in isolation it failed as
pip couldn't find setuptools and 'wheel' to install into the build virtual environment.

● **3 Python templating languages you should (probably) never use** [10]

When reaching for a templating language for writing a Python web application, there are an
abundance of robust solutions.

There are Jinja2, Genshi, and Mako. There are even solutions like Chameleon, which are a bit
older, but still recommended by the Pyramid framework.

Python has been around for a long time. In that time, deep in the corners of its system, it has
accumulated some almost forgotten templating languages that are well worth poking at.

● **How to Speed up Your Python Code** [11]

Always take a good look at your code and algorithms first. Many speed issues can be resolved
by implementing a better algorithm or adding caching.