

Can Gen-AI understand Payments?

04 September 2024 · ai, automation, cloud, deep learning, iso20022, machine learning

When it comes to rolling out updates to large complex banking systems, things can get messy quickly. Of course, the holy grail is to have each subsystem work well independently and to do some form of Pact or contract testing – reducing the complex and painful integration work. But nonetheless – at some point you are going to need to see if the dog and the pony can do their show together – and its generally better to do that in a way that doesn't make millions of pounds of transactions fail – in a highly public manner, in production.



(This post is based on my recent lightning talk at [PyData London](https://www.meetup.com/pydata-london-meetup/) (<https://www.meetup.com/pydata-london-meetup/>))

For the last few years, I've worked in the world of high value, real time and cross border payments, And one of the sticking points in bank [software] integration is message generation. A lot of time is spent dreaming up and creating those messages, then maintaining what you have just built.

The world of payments runs on messages, these days they are often XML messages – and they can be painful to create. You can use libraries like [pyiso20022](https://pypi.org/project/pyiso20022/) (<https://pypi.org/project/pyiso20022/>) or [prowide](https://www.prowidesoftware.com/development-tools/iso20022) (<https://www.prowidesoftware.com/development-tools/iso20022>) to programmatically create payment messages or use some form of template – or one of the commercial test systems available, like UNIFITS (that do a lot more than just generate messages).

As I mentioned in [my last post](https://www.investigatingsoftware.co.uk/2024/08/text-to-swift-making-data-from-prose.html) (<https://www.investigatingsoftware.co.uk/2024/08/text-to-swift-making-data-from-prose.html>) you can also use Gen-AI / Large Language Models (LLM) to create these

messages – and they do a pretty good job of it. They grasp 2 main concepts needed to do the job:

- 1) Understand addresses (Addresses are often in payments messages)
- 2) Recognise payments – as a ‘thing’ in their own right

But I wanted to see how well they did at those tasks – generally. So firstly, I looked closer at addressing. My [previous post's](https://www.investigatingsoftware.co.uk/2024/08/text-to-swift-making-data-from-prose.html) (https://www.investigatingsoftware.co.uk/2024/08/text-to-swift-making-data-from-prose.html) examples were fairly directed – it was clear that I was describing a payment – and I was prompting the LLM with example messages with similar addresses. But what about other addresses:

Here is an example of an address one of my brothers sent me, from French Polynesia:

```
USER
Extract the address and return it in structured form:

CARREFOUR PUNAAUIA
CENTRE COMMERCIAL MOANA NUI
BP 427 PAPEETE, 98713 PAPEETE
```

And here is the parsed / structured version:

```
{
  "Nm": "CARREFOUR PUNAAUIA",
  "PstAdr": {
    "AdrLine": "CENTRE COMMERCIAL MOANA NUI",
    "PstBx": "BP 427",
    "TwnNm": "PAPEETE",
    "PstCd": "98713",
    "Ctry": "PF"
  }
}
```

The LLM did well and there are 2 key points to notice here:

- 1) It has correctly augmented the data with a country code (that was missing) – and it's got the correct one – French Polynesia is a French overseas territory but is also legally a separate country with a country code of 'PF' (not 'FR' for France)
- 2) It struggled with the name of the shopping mall and put this in "AdrLine" – But that actually makes sense – as it doesn't fit easily into to any of the categories that are defined by SWIFT in

its CBPR+ payment scheme. (Building Name is probably the closest). AdrLine is used for 'Hybrid' addresses, where it's been agreed that some section of the message will be - less structured.

Why do we care about these addresses so much?

The world of banking and payments has been moving to new standards for communicating payment information. (The [ISO20022](https://www.iso20022.org/) (<https://www.iso20022.org/>) standard) As part of this staged transition - address information is getting more structured. Previously addresses were largely unstructured and even the 'structured' address used in older SWIFT messages - were mostly free text and pretty limited.

But in the coming months and years - banks and their customers are going to have to structure those addresses better, isolating fields like Post Box (PstBx) and Floor (Flr) from Building Name (you guessed it BldgNm) etc.

You could ask all your customers to rekey the information - not a great customer experience - or rely on some crude ETL and/or regex logic - but that's a project and maintenance overhead in itself - and will need to be adapted for each jurisdiction your customers & beneficiaries are located in (addresses vary greatly around the world, e.g. most Japanese streets are not named - Korea has 2 systems of addressing, Spanish house numbers go after the street name etc)

But do LLMs really understand payments?

Going back to the payment message, addresses are handled reasonably well - what about the concept of a payment? My initial example was 'free text' or prose, but still recognisable as me describing a payment.

Can I get a Large Language Model / Gen-AI to recognise payments from a news article?

If we take this article about a baseball player who was defrauded by his translator:

Major League Baseball

[+ Add to myFT](#)

Baseball star Shohei Ohtani's former interpreter charged with bank fraud

Prosecutors say Los Angeles Dodgers player was the victim of a \$16mn theft



The justice department said Mizuhara made "unauthorised transfers" worth more than \$16mn from November 2021 to January 2024 from Ohtani's account. "The transfers from this bank account allegedly were made from devices and IP addresses associated with Mizuhara," the DoJ said.

We can see that we can extract structured details about the payment in JSON format (a format easily readable by software, like the XML messages I mentioned above).

This took some trial and error, as for example, news articles tend to be summaries and rather than specifying exact 'value dates' for a payment, tend to provide an approximate date range.

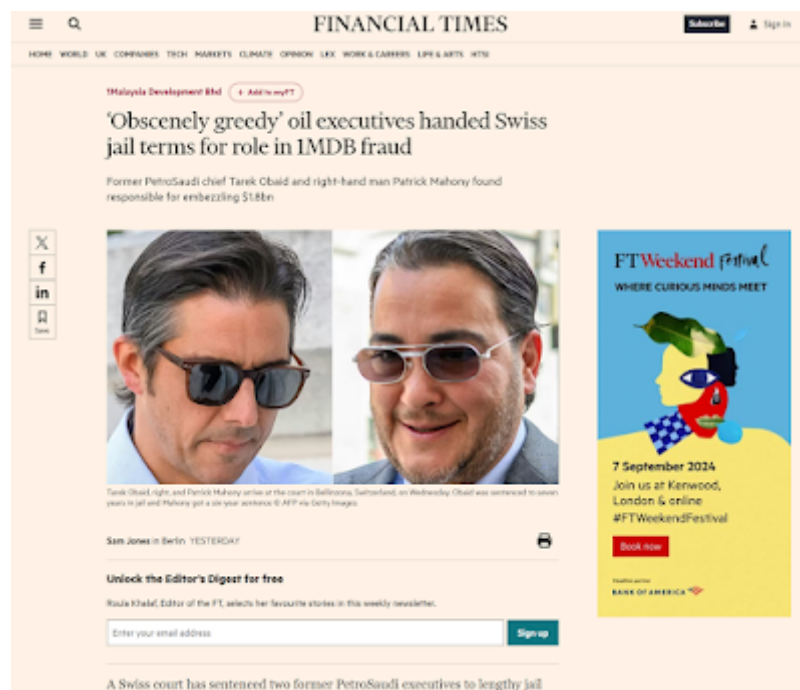
```
{
  "payments": [
    {
      "amount": "16,000,000",
      "currency": "USD",
      "debtor": "Shohei Ohtani",
      "creditor": "Ipeei Mizuhara",
      "date_from": "2021-11-01",
      "date_to": "2024-01-01",
      "details": "Unauthorized transfers to cover gambling debts"
    }
  ]
}
```

But here we can see the dates and values have been extracted correctly, though I might question whether it should read the 1st Jan 2024, or the last day of that month - though the original article is a little vague.

Let's try something messier...

But that's a fairly simple example, what about a more complex fraud with multiple payments, payment types and not just one guy and his translator.

So here we analysed an article about how some people defrauded the sovereign wealth fund of Malaysia (1MDB).



Let's try something messier...

As you can see, we get a more complex output - and on closer inspection I could see it covers all the payments involved during and after the fraud. Here they are in JSON:

```

{
  "amount": "830,000,000",
  "currency": "USD",
  "debtor": "IMDB",
  "creditor": "Fake joint venture (controlled by Taek Jho Low)",
  "date_from": "2010-01-01",
  "date_to": "2010-12-31",
  "details": "Islamic Loan to fake joint venture"
},
{
  "amount": "192,000,000",
  "currency": "USD",
  "debtor": "Assets frozen by Swiss authorities",
  "creditor": "Swiss authorities",
  "date_from": "2018-01-01",
  "date_to": "2023-04-25",
  "details": "Assets traced to IMDB"
},
{
  "amount": "3,900,000,000",
  "currency": "USD",
  "debtor": "Goldman Sachs",
  "creditor": "Malaysia",
  "date_from": "2020-01-01",
  "date_to": "2020-12-31",
  "details": "Settlement for role in IMDB scandal"
}
]

```

```

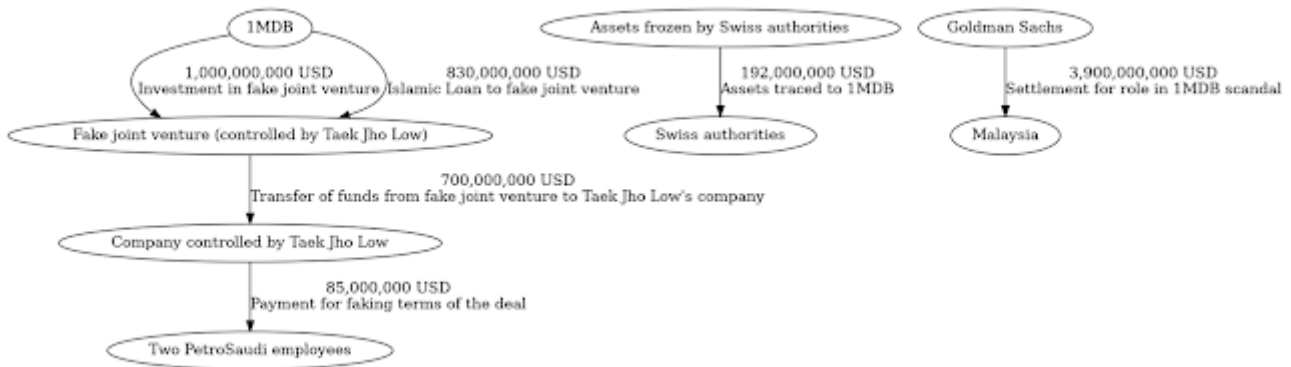
{
  "payments": [
    {
      "amount": "1,000,000,000",
      "currency": "USD",
      "debtor": "IMDB",
      "creditor": "Fake joint venture (controlled by Taek Jho Low)",
      "date_from": "2009-08-01",
      "date_to": "2009-12-31",
      "details": "Investment in fake joint venture"
    },
    {
      "amount": "700,000,000",
      "currency": "USD",
      "debtor": "Fake joint venture (controlled by Taek Jho Low)",
      "creditor": "Company controlled by Taek Jho Low",
      "date_from": "2009-12-01",
      "date_to": "2009-12-31",
      "details": "Transfer of funds from fake joint venture to Taek Jho Low's company"
    },
    {
      "amount": "85,000,000",
      "currency": "USD",
      "debtor": "Company controlled by Taek Jho Low",
      "creditor": "Two PetroSaudi employees",
      "date_from": "2009-12-01",
      "date_to": "2009-12-31",
      "details": "Payment for faking terms of the deal"
    }
  ],
}

```

Why do we want to extract this data into JSON or XML?

While crude tools that match on keywords or even semantic meaning (e.g. a search for Stealing matches Theft). What we often want to do is do the next step, act on the data – what fundamental knowledge of a fraud / payment is in the text(s)? To start with – we can visualise the connections between the payments.

Our structured was then processed again by our LLM, which can locate the connections between the payments – and generate graphical data (here I configured the system to use ‘dot’ language and [GraphViz](https://graphviz.org/) (<https://graphviz.org/>))



The output is useful, and improves the legibility, and makes it much easier to quickly grasp the scope and range of payments. Conceptually, we've gone from:

A free text news article - to - A list of data about payments - to - a directed graph of those payments, presented in a user friendly graphic.

Overall, I'm impressed with the power of the Gen-AI tools available. They don't work well on these data and prose 'out of the box' - if you just ask Chat-GPT for a payment initiation message - you will get nonsense. But with the right tools, data and prompting they add considerably to the capability of modern software to generate and analyse payments.

What AI models did I use?

I found smaller versions of the LLMs e.g. Llama 3.1 8B didn't do great, but models bigger than 20B parameters did work quite well. The most capable model was GPT4o but the option to provide an output schema in GPT4o-mini was very useful. But the fact that open source models like Llama and Gemini worked well (in their larger versions) is a significant point - why? Well as they are open source - you can run them on-premises without the need for cloud systems.

Running 'on-prem' is a common ask from banks and financial institutions - due to regulatory and in house security concerns about customer and financial data security. Hosting an open source GPT4o level model is now possible on premises with Llama 3.1 405B (the biggest version). That combined with the above analysis, generation and processing tools could both improve system maintenance and help with investigations, Recs (reconciliation of payments) or even AML (Anti Money Laundering).