Culinária Computacional: Uma Análise da geometria de ingredientes e receitas

Introdução

Neste projeto tenho o intuito de estudar as relações geométricas entre ingredientes e receitas utilizando como base um banco de dados com mais de 2 milhões de receitas. Entretanto a ideia original que resultou neste conceito foi um algoritmo de recomendação de receitas baseado em ingredientes escolhidos, algo pessoal para mim como programador já que engloba duas das minhas maiores paixões, cozinha e programação. Este conceito, como foco principal , foi deixado de lado por conta da falta de necessidade do uso de ferramentas estudadas no curso para sua realização. Porém a partir do banco de dados, modelagem, e tratamento de dados que havia criado foi possível expandir o código para esta nova direção com muito mais potencial e complexidade. Resultando em um programa capaz de identificar a distância entre receitas e ingredientes, permitindo um melhor algoritmo de recomendação e uma nova perspectiva nesta área de culinária computacional. Por fim o programa final é capaz de receber um ingrediente ou receita e retornar o ingrediente ou receita mais próximo utilizando mínimos quadrados alternados com um posto de X para uma matriz original de 1000x1097 tal matriz criada utilizando 1000 entradas de receitas, e todos os ingredientes presentes nas mesmas, obtidas no banco de dados utilizado. Em outras palavras, indicando a receita mais parecida ou o ingrediente com usos mais similares ou que combinaria com o ingrediente de entrada em uma receita.

Ferramentas básicas

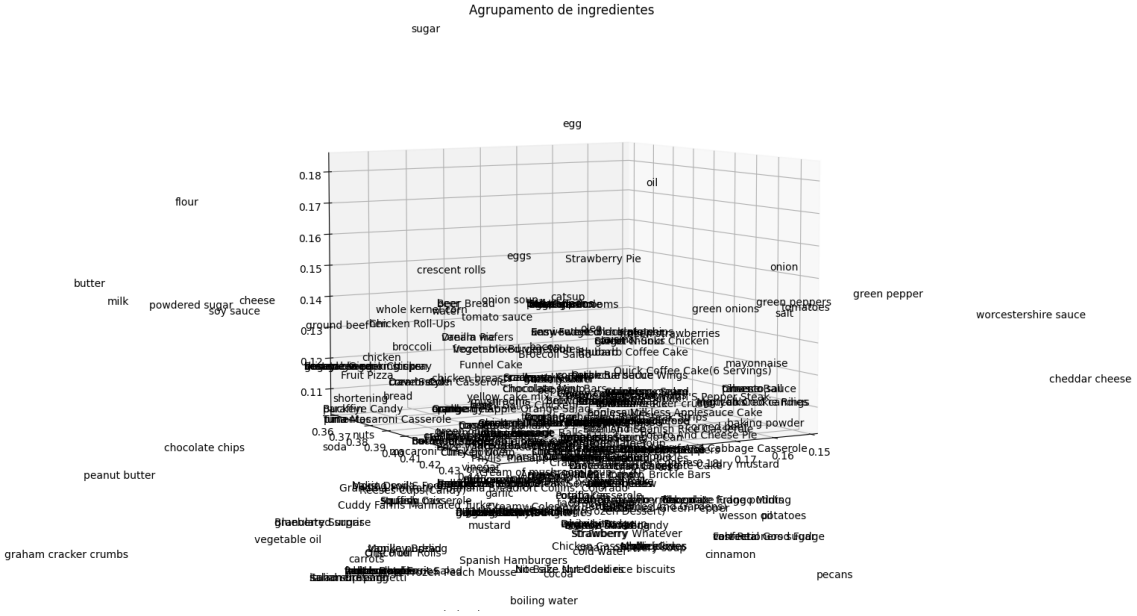
Para este desenvolvimento foi utilizado o Kaggle para pesquisar e baixar um banco de dados adequado para o projeto. A linguagem de programação escolhida foi o Python 3 devido a simplicidade e experiência prévia com arquivos no formato .csv, com o auxílio das bibliotecas Pandas para a leitura do dataset, Numpy para o trabalho avançado com arrays e o MatPlotLib para a exibição de gráficos.

Metodologia

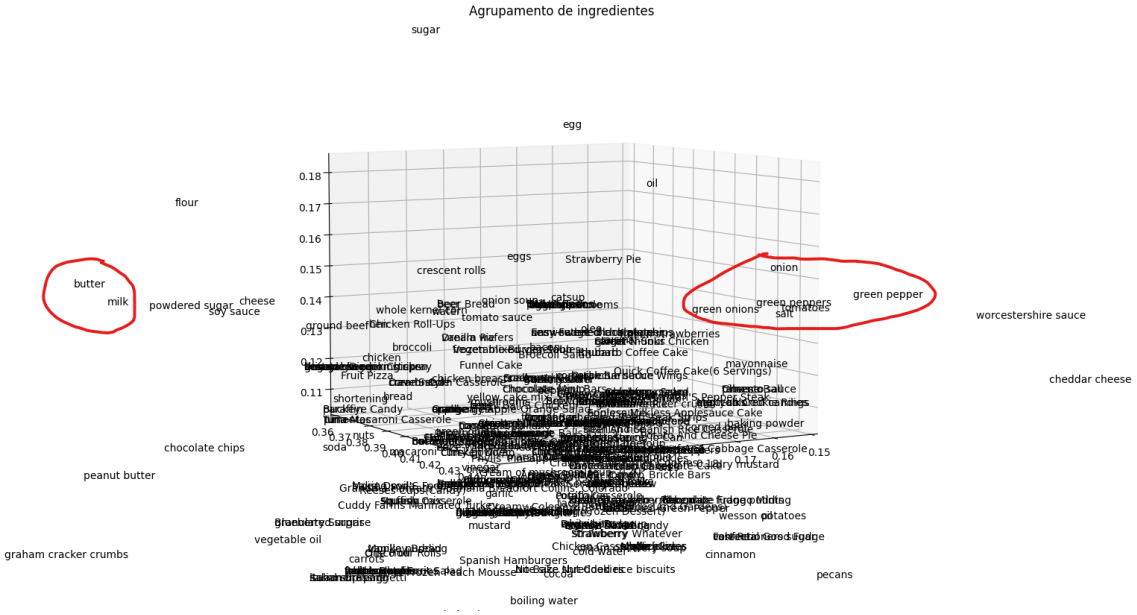
Como mencionado na introdução o método mais utilizado no código foi o ALS (alternating least square) ou mínimos quadrados alternados, com o intuito de decompor a matriz receitas por ingredientes em duas matrizes receitas por k e k por ingredientes com k sendo 3 nas exibições gráficas em R3 e valores maiores nas funções de aproximação visando menor erro no método. Além disso, a modelagem da matriz receitas por ingredientes foi feita utilizando o banco de dados que continha cada receita e seus ingredientes, e é composta por 0s e 1s indicando se cada ingrediente está presente na receita. Além destes métodos foi feito uma análise do erro do ALS para identificar qual k ideal, o que gerou as matrizes B e C, tal que A - BxC era mais próximo de zero, iterando o método para Ks de 1 até 1000 (q quantidade de linhas da matriz original).

Resultados

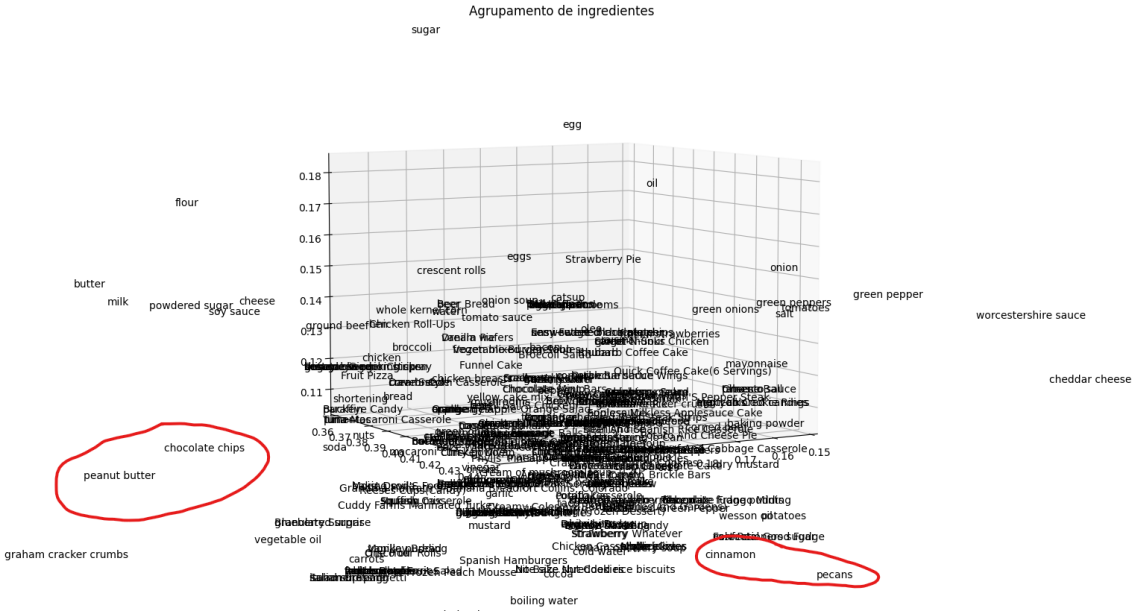
Apesar de os conceitos e os métodos usados parecerem corretos obtive alguns problemas com o resultado, majoritariamente na análise de erros, o último processo feito e o que demandava mais tempo computacional criando um limite de testes do código possíveis.
Em resumo o ALS não parecia convergir, ficando estável no valor inicial baseado no chute da primeira iteração, com um erro médio, ao variar o posto, ficando em 83 e o melhor resultado 80 para postos próximos de 890. Além disso, o fenômeno mais estranho que infelizmente não consegui resolver ou identificar a causa foi que o método utilizando postos próximos a 1000 para uma matriz 1000x1096, que deveriam gerar um erro ínfimo estavam fazendo o exato oposto, com um erro completamente fora da curva altíssimo.
Por fim devido ao tempo que cada iteração do código levava para calcular os erros optei por aceitar o erro da primeira iteração do método e usar um posto baixo visto que a melhora do erro em menos de 6% não justificava o aumento do tempo computacional para o posto 890 e o número de iterações no ASL não influenciava o resultado final.
Entretanto, mesmo com as falhas nesta etapa, o código consegue performar na feira e em vários testes com resultados coerentes e interessantes. Obviamente não temos como ter certeza da precisão dos resultados já que o erro não foi otimizado e foi possível forçar o programa a dar respostas ruins. Porém no geral o objetivo foi alcançado de alguma forma como podemos ver em alguns dos exemplos:
Aqui temos um plot em 3D da dispersão de ingredientes e algumas interpretações:



Temos alguns casos óbvios de proximidade de ingredientes baseado no tipo, laticínios e legumes para temperar por exemplo:



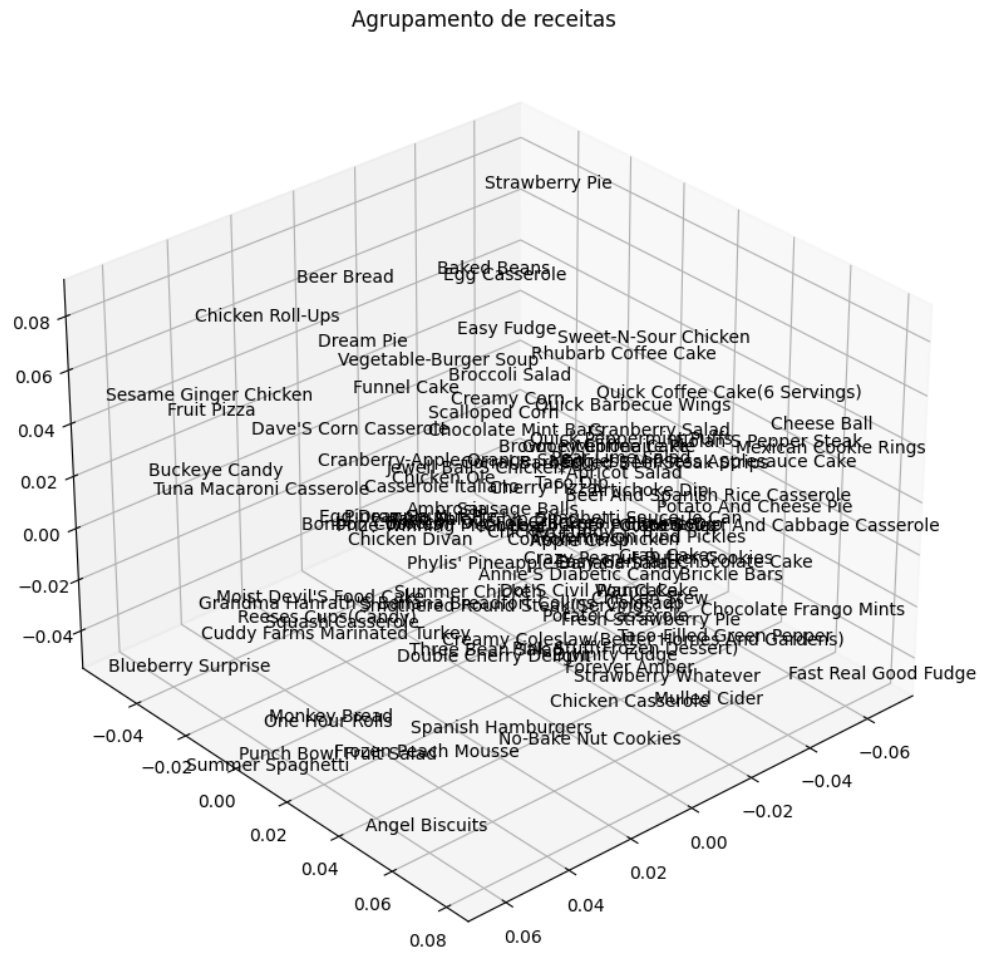
Temos também casos de proximidade por uso similar, nos dois grupos marcados vemos ingredientes utilizados em cookies e receitas de nozes com frequência.



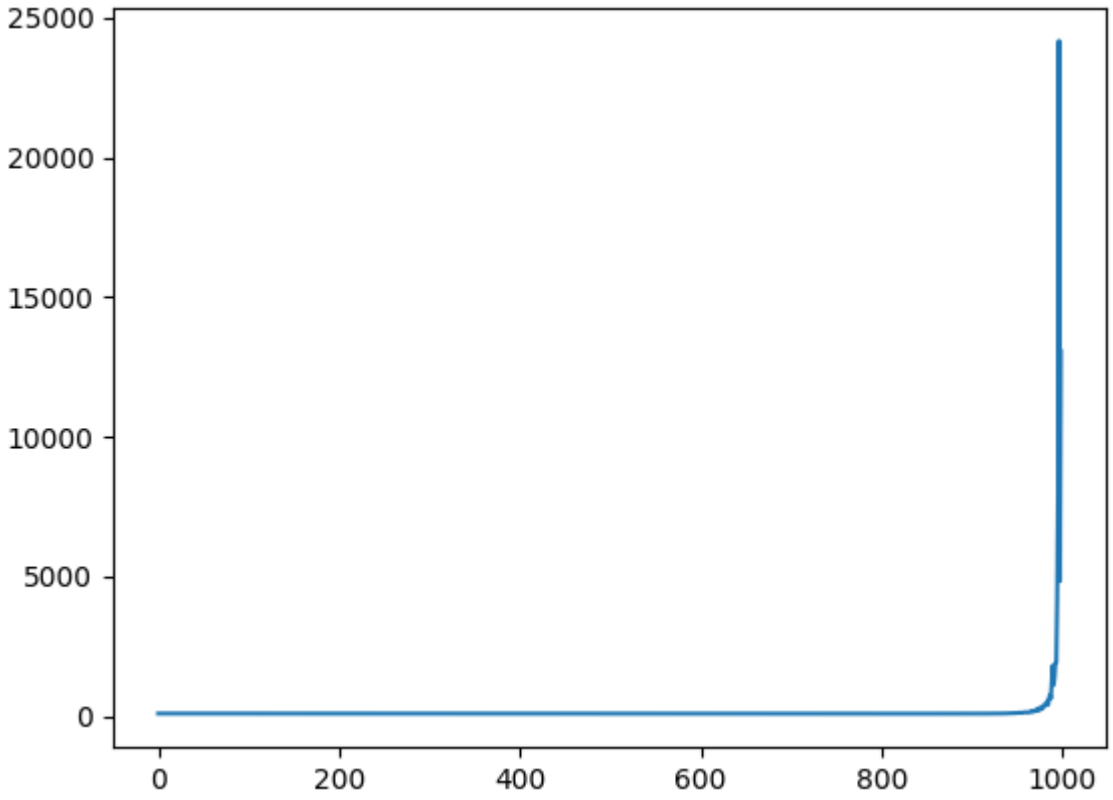
É importante ressaltar que este gráfico é de difícil análise já que é em 3D e a perspectiva pode dar respostas incorretas. Por conta disso temos um código que indica o ingrediente mais próximo em cada caso:

```
>>> ingredienteProximo('egg')
0.06462611068695309
cream of celery soup
>>> ingredienteProximo('cinnamon')
0.046044735068327944
raisins
>>> ingredienteProximo('sugar')
0.2031826151378194
butter
>>> ingredienteProximo('milk')
0.07726757400340654
pineapple
>>> ingredienteProximo('chocolate chips')
0.02459734500231009
cocoa
...
```

Nestes exemplos vemos acertos claros como 'chocolate chips' e 'cocoa', alguns enganos gerados pela perspectiva como 'milk' e 'pineapple' e alguns erros mais claros como o fato de que 'egg' e 'eggs' não serem os mais próximos entre si.
Já no agrupamento de receitas temos um gráfico ainda mais difícil de visualizar por conta de os nomes das receitas não serem intuitivos em relação ao seus ingredientes:



Algumas interpretações e observações considerando os resultados podem ser feitas. Dentre elas é importante notar que como o código proximoIngrediente() e o proximoiReceita() gera uma matriz sempre que tenta encontrar um ingrediente próximo, e portanto a resposta tende a variar, isso porque existe uma falha no algoritmo de mínimos quadrados e consequentemente uma inconsistência. Este problema, somado à falta de interatividade das receitas faz com que o código de proximidade de receitas não tenha resultados muito óbvios ou consistentes.
Por fim temos o gráfico de erro do nosso método ALS indicando suas falhas:



Podemos ver que a mudança de posto não afeta o erro de nenhuma forma relevante, o que claramente é um erro, e ao chegar próximo do posto onde o erro seria o menor possível teve um crescimento tendendo ao infinito. A análise desta questão em particular é feita mais a fundo na conclusão.

Conclusão

Este projeto mesmo com as suas falhas óbvias foi uma atividade muito satisfatória e ao final do desenvolvimento, da feira e das conversas que tive sobre ele, minha única conclusão válida é que foi um sucesso parcial. A ideia original de criar algo que incitasse o interesse na computação científica foi completamente alcançada, o meu entendimento sobre os métodos utilizados foi alterado com uma nova perspectiva do uso prático das teorias aprendidas e no final resultados aceitáveis foram obtidos.
Acredito que com mais tempo o erro seria identificado e seria possível afirmar com certeza que os resultados são aceitáveis. Para fins acadêmicos no momento minhas hipóteses para

o erro são a má execução do algoritmo, talvez pelo uso do python e não do julia, ou alguma questão de sintaxe que passou despercebida, visto que a ideia realmente parece estar correta pelas minhas pesquisas e testes fora do código.
Além destas questões acho importante mencionar que o uso de apenas 1000 entradas em um banco com mais de 2 milhões foi uma necessidade de hardware que talvez fosse possível contornar utilizando algumas ferramentas de otimização. Esta redução em um tipo de informação tão heterogênea criou resultados que num vácuo fazem sentido, porém seriam considerados incorretos na vida real, principalmente se tratando das receitas. Tenho certeza de que com mais dados os resultados seriam ainda melhores e mais interessantes.
Por fim avaliando possíveis direções para expansão do código assim que finalizado seria interessante, utilizando um algoritmo de classificação, classificar as receitas ou os ingredientes e utilizar esta nova informação para interpretar os eixos dos gráficos e relevância de cada ingrediente na classificação final (doce, salgado, confeitaria...). Outro caminho que tenho bastante interesse de perseguir é utilizando a coluna de métodos de preparo do banco de dados criar uma nova matriz Receitas por Métodos e tentar trabalhar com mais informações podendo no final ser capaz de gerar receitas completas com ingredientes e instruções ou descobrir quais os métodos de preparo mais comuns para cada ingrediente.

Referências

Banco de dados:
https://www.kaggle.com/datasets/wilmerarltstrmberg/recipe-dataset-over-2m
Código de mínimos quadrados alternados usado em sala:
https://colab.research.google.com/drive/1FumrxpICaV-A7gCgXnUphuQlucYFGwyX?authuser=1#scrollTo=ldfMp62B_hh0
Referencias da biblioteca do numpy: https://numpy.org/doc/stable/reference