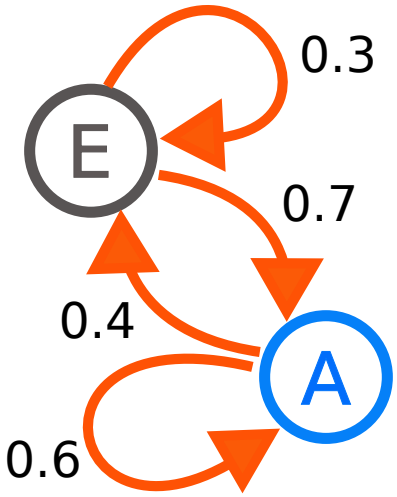
Markovketens

Een Markovketen (“Markov chain” in het Engels) beschrijft een systeem dat zich tussen verschillende toestanden beweegt. Hierbij is de volgende toestand slechts gebaseerd op de huidige toestand.



Figuur 1: Een simpele Markovketen

(Bron: https://en.wikipedia.org/wiki/File:Markovkate_01.svg, licentie: CC-BY-SA of GFDL)

Je kunt een Markovketen beschrijven als een gerichte graaf, waarbij de knopen de verschillende toestanden representeren en de pijlen de mogelijke overgangen representeren. De gewichten van de pijlen representeren de kans dat het systeem zich over die pijl 'beweegt' naar de volgende toestand. In *figuur 1* zien we een simpele Markovketen. Er zijn twee toestanden, E en A, en bij elke toestand twee pijlen: een pijl met een kans dat het systeem naar de andere toestand gaat en een pijl met een kans dat het systeem in dezelfde toestand blijft.

Markovketens voor namen

Je kunt het genereren van tekst of gewoon een naam ook beschrijven als een Markovketen, elke toestand is dan een letter en dan krijg je uit de volgorde van deze toestanden een naam. Verder heb je natuurlijk nog een eind en begin toestand nodig, want anders weet je niet wanneer je moet stoppen en bij welke staat je moet beginnen, deze toestanden noemen we EIND en BEGIN respectievelijk.

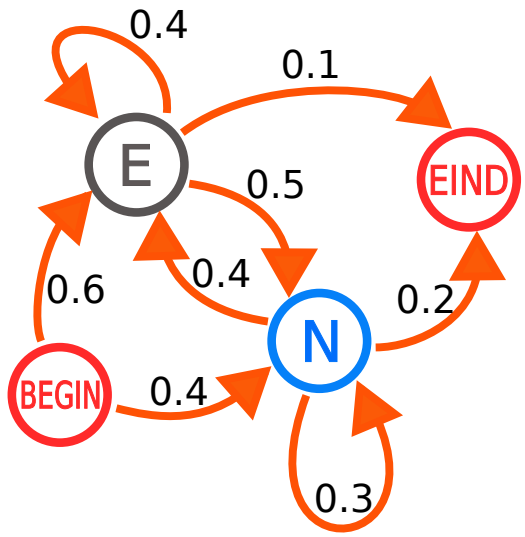
Laten we beginnen met een simpel voorbeeld waarin we alleen de letters 'e' en 'n' gebruiken, zie *figuur 2*. Dit is al iets ingewikkelder dan *figuur 1*, maar het principe blijft hetzelfde.

Laten we beginnen bij BEGIN, vanuit daar zijn er twee mogelijkheden, we kunnen naar de 'e' of de 'n'. De kans dat we naar 'e' gaan is 0,6 en de kans voor 'n' 0.4. Dit hebben we zo gekozen, omdat we denken dat namen vaker met een 'e' beginnen dan met 'n'. Dezelfde soort keuzes hebben we moeten

maken voor de pijlen vanuit 'e'. Er is een kleine kans dat dit het einde van het woord is en een grotere kans dat we nog een keer 'e' krijgen en een nog grotere kans dat we nu 'n' krijgen.

Al dit soort keuzes goed maken wordt al snel erg lastig, maar gelukkig zijn er manieren om een Markovketen als het ware te trainen met voorbeeldnamen, daarover komt later meer. Eerst gaan we nu bekijken hoe we met een dergelijke Markovketen een naam genereren.

We beginnen bij BEGIN, dan besluiten we willekeurig op basis van de kansen van de pijlen of we naar 'e' of 'n' gaan, laten we aannemen dat we deze keer naar 'n' gaan. Tot nu toe hebben we de naam “n”, vanaf hier kunnen we naar 'e', 'n' of EIND, we kiezen gebaseerd op de kansen willekeurig voor de 'e'. Nu hebben we de naam “ne”, nu kiezen we willekeurig uit 'e', 'n' en EIND met als uitkomst 'n'. Nu hebben we de naam “nen”, nu kiezen we wederom en krijgen we 'n'. We hebben nu de naam “nenn”, we kiezen nog een keer en krijgen EIND. Nu zijn we klaar en hebben we de naam “nenn” gegenereerd. Dit is natuurlijk niet een erg mooie naam en dat is grotendeels te wijten aan het feit dat we alleen de letters 'e' en 'n' gebruikten. Maar ook aan het feit dat de kansen die ik aan de pijlen heb gegeven niet optimaal zijn en natuurlijk ook aan ongeluk.



Figuur 2: Een iets ingewikkeldere Markovketen, gebaseerd op figuur 1

Als we dit proces nog een paar keer herhalen krijgen we elke keer een andere naam, dit proces doen we natuurlijk niet handmatig, want dat zou erg foutgevoelig en tijdrovend zijn, we hebben hiervoor een programmaatje geschreven, wat bijgevoegd is als *simple-markov.scm*. Een paar uitkomsten van dit programma zijn: “eeeeenn”, “ee”, “enenn”, “neneen”, “enen”, “eeenenenenennn”, “eene”, “enn”, “n” en “neeneneenen”.

Markovketens trainen

Het is lastig om handmatig de data voor een grotere Markovketen te bedenken. Niet alleen is het lastig om de bedenken wat de kans is om van de éne letter naar de ander te gaan, ook moet je dit heel vaak doen. Daarom is het handig om een Markovketen als het ware te trainen op voorbeeld namen. Hierbij haal je uit een zo groot mogelijke set van voorbeeld data de kansen waarmee elke letter elke andere letter opvolgt.

Voor dit doel hebben we een lijst van eiland namen samengesteld vanaf Wikipedia, die we daarna met behulp van wat programma's en ook gedeeltelijk handmatig schoon hebben gemaakt (dus bijvoorbeeld de coördinaten die er op Wikipedia achter stonden). Deze lijst is bijgevoegd als *names.txt*. Een aantal voorbeeld namen uit deze lijst zijn: “Loreto”, Porvenir”, “Tigre”, “Yaguas”, “Muñoz”, “Pernambuco”, “Margarita”, “Tigrera”, “Venezuela”, “Aduché”.

Als we hiermee een naïeve implementatie van een Markovketen voor namen mee trainen, krijgen we namen zoals: “Etibelangocharst”, “Lamarnd”, “Wochiouastr”, “Manngulledug”, “Murtascckbutégrockore”, “Webbbbamp Isayrna”, “Trocas”. Sommigen van deze namen, zoals

“Trocas”, zijn redelijk goed, maar de meesten zijn zeer ongeloofwaardig. Zo heeft “Webbbbamp Isayrna” 4 b's achter elkaar. We zouden dit soort problemen kunnen verminderen door de namen te gaan controleren aan de hand van bepaalde criteria, zoals “niet meer dan twee keer dezelfde letter”, maar het is veel werk om alle mogelijke restricties zelf te bedenken. We willen liever dat de Markovketens zelf beter van de voorbeeldnamen leren zodat we betere namen krijgen.

Omdat de naïeve implementatie maar 1 letter 'onthoud' is er de mogelijkheid om meerdere keren dezelfde letter achter elkaar te krijgen of andere onmogelijke combinaties zoals “scckb” in “Murtascckbutégrockore”, terwijl deze in het voorbeeld niet eens voorkomen. Om dit te verhelpen kan je elke toestand niet laten afhangen van de laatste letter, maar van de laatste N letters. Bijvoorbeeld als je tot nu toe “Sch” hebt en N is 2, dan is de huidige toestand “ch” en kijk je niet naar alle de frequenties waarmee de letters alleen na de 'h' komen, maar naar de frequenties waarmee letters na de combinatie “ch” komen. Vervolgens kies je op basis daarvan de volgende letter, bijvoorbeeld de 'o'. Nu is de huidige toestand “ho” en kies je op basis daarvan een volgende letter, en zo ga je verder.

Hoe hoger N, hoe meer de gegenereerde namen op de voorbeeldnamen beginnen te lijken. Met een N van 5 en als trainingslijst *names.txt* (dat 1886 namen bevat) is zijn vrijwel alle namen in de output precies overgenomen uit de trainingslijst. Als we dat hadden gewild hadden we gewoon direct willekeurig uit de lijst kunnen kiezen! Dit probleem ontstaat omdat met een N van 5 de combinaties zoals “Tiriz” meestal maar één keer voorkomt in de trainingslijst, waardoor er maar één mogelijke volgende letter is, namelijk de 'i'. Dan heb je de toestand “irizi”, waarna alleen de 's' kan komen en met de toestand “rizis” die dan ontstaat kan je alleen de naam eindigen. Zo ontstaan telkens namen die precies in de trainingslijst voorkomen.

We willen dus een N kiezen waarmee de namen er zo echt mogelijk uit zien, maar zo min mogelijk precies hetzelfde zijn als voorbeeldnamen. Voordat we kiezen welke N hiervoor het beste is, bespreken we eerst onze implementatie van dit principe in C++.

Implementatie

>> nog te doen <<