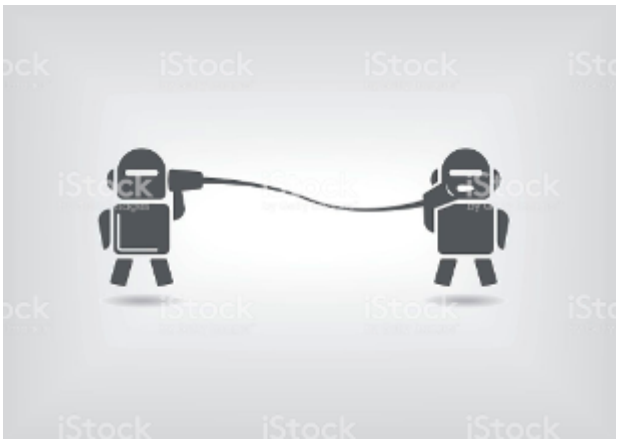
Ramon Garcia Seuma Follow
Aug 7, 2019 · 10 min read · Listen

Dynamics for language conventions: Naming Game

It is possible to describe how in an interconnected network of agents, without targeted intervention, can appear a set of shared conventions spontaneously? Here I show you one model that allows us to do that: the Naming Game.



The game is a minimal model and it was firstly devised for an experiment involving a group of robots, to see whether they could develop an own dictionary for the communication among them with no need of human intervention. You can think on it also as a theoretical framework, a basis for describing the emergence of shared keys for encrypted communications, or in a human sphere it can describe the basic aspects of the recent phenomenon of collaborative tagging or social bookmarking on popular web portals like Del.icio.us or Flickr.

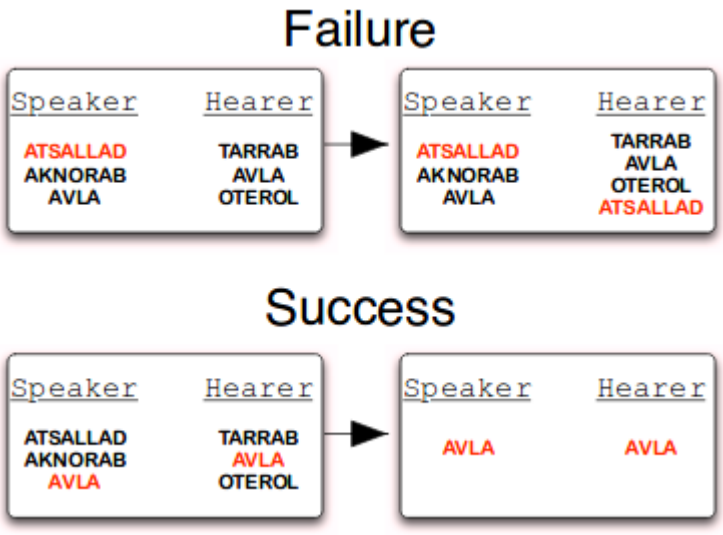
The model is intended to help me introduce you the field of Complex Systems, the agent-based modeling mindset and the kind of analysis taken out from Physics that allows us to, more than analyze certain phenomena (in social, economical or biological networks, for example) out from large amount of data, create models to produce this data and grasp the inner basic laws that describe their dynamics. Thus, it gives us a broader vision of those phenomena, the ability to imagine situations that could arise out from it, by the exploration of the space parameter, so to detect phase transitions and treat this systems as we do in Physics (like liquid water going to gas as we heat it).

concepts out from Physics, as statistical physics and thermodynamics, to a broad range of different systems in biology, engineering, sociology and economics.

The main idea behind it is that many systems display emerging collective behaviors and self-organization that can not be explain from the individual properties of the agents the system is composed of. The complex topology (who can interact with whom) of the inner network and the interaction within its agents makes that collective phenomena appear. In biology, you can think in the swarming behavior of bird flocks, fish schools, insects and bacteria (or just people in a demonstration). In economics, in the distribution of wealth in a modern society and price formation dynamics, for example. In social systems, in the formation and spreading of choices and opinions, such as in political elections. There are a lot of examples of situations that can be viewed within this framework; here I present just one model to reproduce how in a certain network the agents can agree and achieve a global consensus in how to design some objects perceived by all them.

The Game

The game has a very simple dynamics. Our system consists of a number of agents, say N, that can be connected differently, and all have its particular dictionary of words (stored in their memories). Then we have objects, and each object has its own entry in each dictionary for designing it in different ways. We can limit our model to have just one object, as we get qualitatively the same dynamics.



Two possible interactions between the speaker and hearer.

We can now think as everyone being connected with everyone: this is called a *complete graph* in graph theory. Then, the dynamics performs in the following way: at each time step we select one

only the success word. If the hearer has not the word in the dictionary, we have a failure, and he just adds the word at the end of the dictionary. In principle, we don't know what to expect to get as an output for the game, since we have a large number of agents so the system in non-linear, thus not very intuitive. We can see the algorithm summarized in python code below.

```
def step(N, neighs, inventary):
    success_rate = 0
    for i in range(N):
        speaker = np.random.randint(N)
        hearer = np.random.choice(neighs[speaker])

        #if the dictionary is empty, create a word randomly
        if len(inventary[speaker]) == 0:
            inventary[speaker].append(np.random.choice(10**5))

        definicion_speaker = np.random.choice(inventary[speaker])
        success = False
        for objs in inventary[hearer]:
            if objs == definicion_speaker:
                success = True

        if success:
            inventary[speaker] = [definicion_speaker]
            inventary[hearer] = [definicion_speaker]
            success_rate += 1

        if not success:
            inventary[hearer].append(definicion_speaker)

    success_rate /= N
    return inventary, success_rate
```

Initially, each one of the dictionaries are empty, but we create a word randomly when a new hearer enter in action. As you can see, my "words" are numbers from *0* to 10^5*-1*, so a most accurate designation would be just "description", as we need to have simply a way to codify a particular designation of an object. I select such a large number for not being in risk, even when N is large, of having coincidences between yet non-interacted dictionaries. The risk is that if we had more than one object, we could have homonym words, and this would affect the dynamics (indeed this is one variant of the game we could study with this model).

The code above shows one time step of the game, and this time unit is defined to be (as in most agent-based dynamics) equal to the performance of N games, so when in average all agents have played the game at least once, we advance the time one unit.




```
def simulation(N, macrosteps):
    neighs = extract_allwithall(N) # for complete graph
    #neighs = extract_neigh_square(N) # for 2D lattice
    #neighs = extract_small_world(N,4,0.08) # small world
    #neighs = extract_scalefreeBA(N,2) #scalefree barabasi-albert

    inventary = [[] for i in range(N)]

    t = 0
    N_w, N_d = counts(inventary)
    total_words = [N_w]
    different_words = [N_d]
    successes = [0]

    for i in range(macrosteps):
        t += 1
        inventary, success_rate = step(N, neighs, inventary)
        successes.append(success_rate)
        N_w, N_d = counts(inventary)
        total_words.append(N_w)
        different_words.append(N_d)

    return total_words, different_words, successes
```

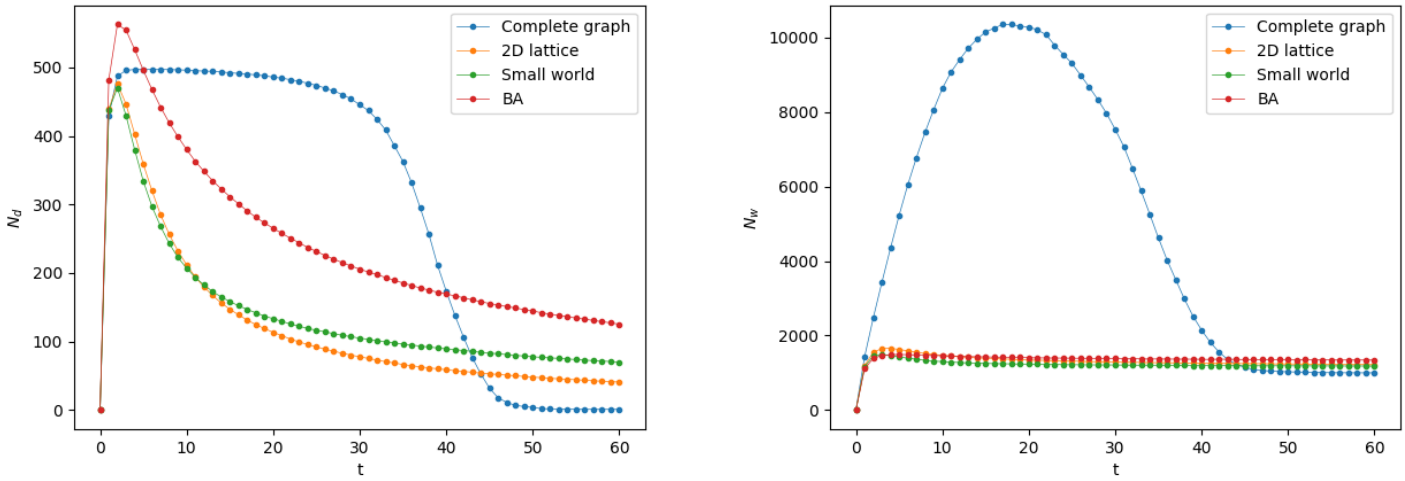
For a visual and simple demonstration of how the dynamics evolve, you can check out a basic interactive simulation here https://ccuskley.github.io/colornaming.html.

Then we proceed to run simulations and doing statistical averages. The clue here is that we have an ***stochastic process***, so at each simulation we are going to have fluctuating outputs around certain value. This mean value is going to be the most valuable information of our simulations. So we just need to take averages of many simulations. Important quantities to analyze are the total words length, the number of words at the end of a game and the mean success rate for our players.

I show you my results of this quantities, for different kinds of networks, and how we can extract useful information out of it. We can generalize the network, and now imagine that not every agent (called *node* in graph theory) can communicate with each other, as this is a more realistic situation. The input for describing such a network is the neighbors list, that describes who is connected with whom. This links between nodes are called *edges* in graph theory. So you can imagine basically a network of points connected by lines. Then, when we pick up a speaker, we search randomly one of its connected nodes to act as a hearer. Initially I use this kinds of basic graphs:

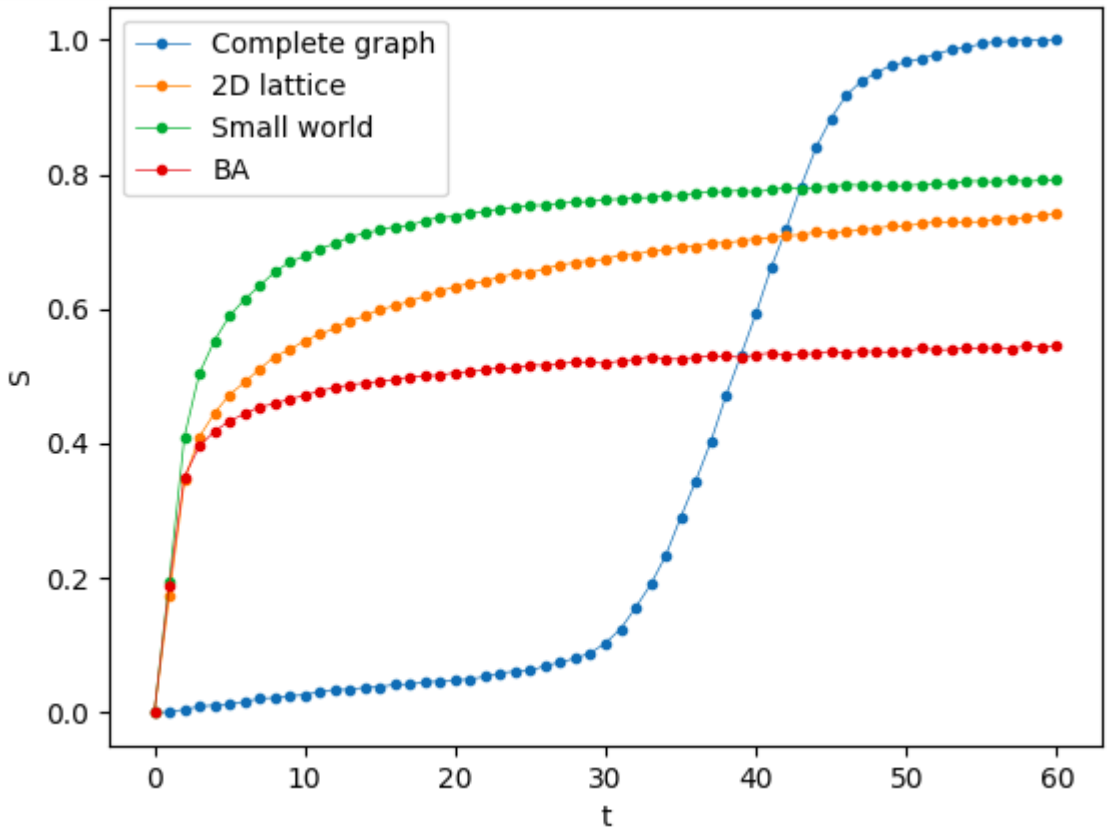



- Small world: nodes are connected with its 4 immediate neighbor as in the 2D lattice but then with a certain probability reconnected with a distant node. So we have mixed connections with near nodes and distant ones.
- Scale-free network: Few nodes have lots of connections, called hubs, while the majority have few ones. This is one main feature of the networks encountered in many natural systems, mainly in social ones. Think for example in the WWW network, pages like Google of Facebook have millions of connections and then millions of pages like my blog having few ones. I particularly use one kind of scale-free network, the Barabási-Albert graph.



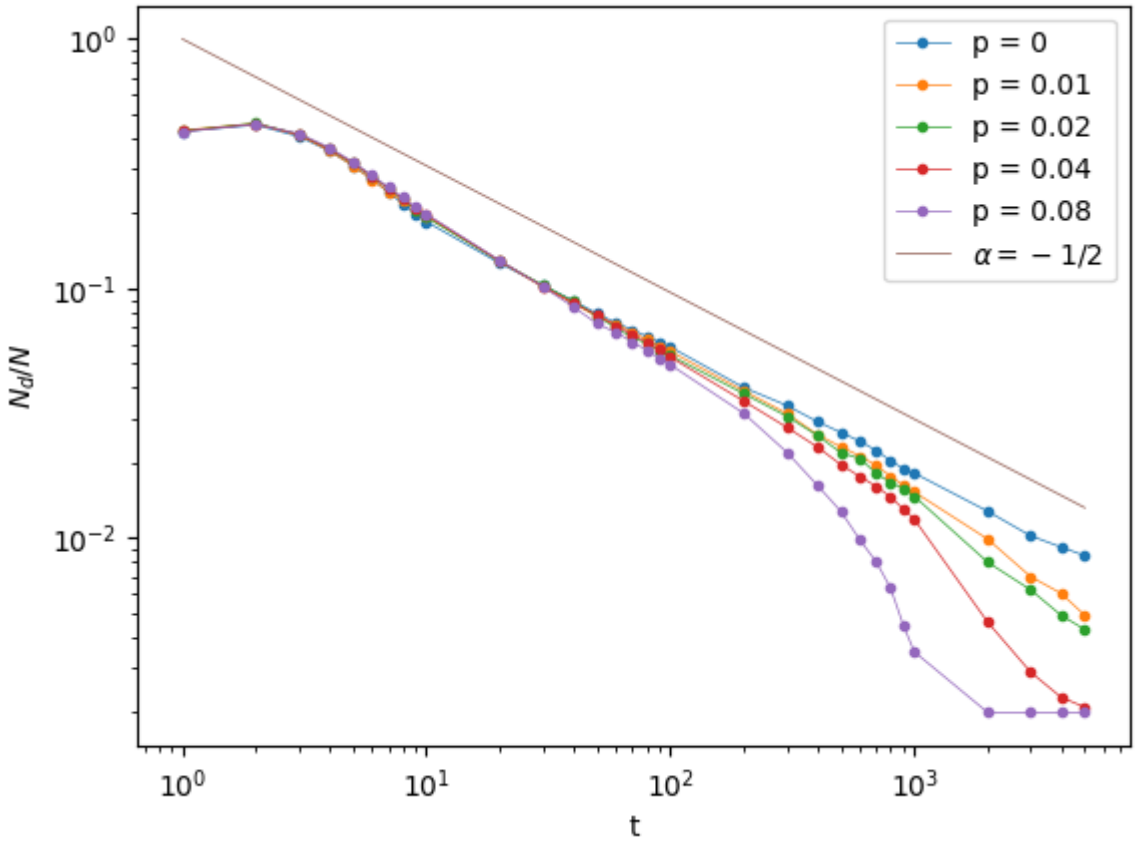
Number of different words (left) and total words summing all dictionaries (right) vs time, in networks of N = 1000 nodes, averaged over 100 simulations.

As we can see in the previous figures, in the complete graph much more quantity of words are created. That means that if we wanted to implement the dynamics in a network of sensors with encrypted keys, it would be less efficient and require more memory if we connected every node with each other.



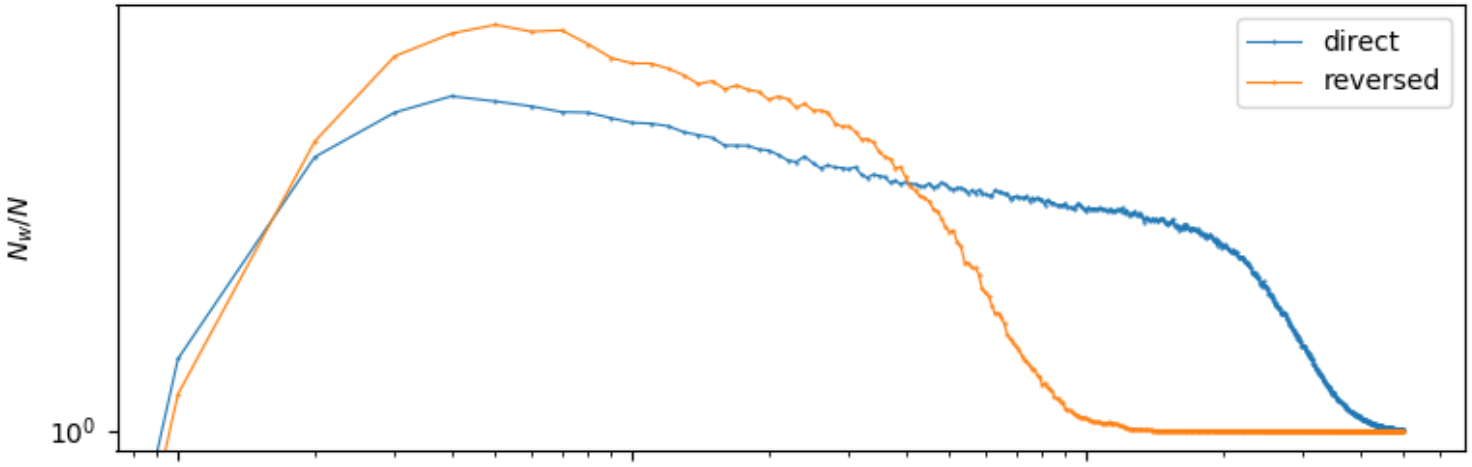
Succes rate in networks of N = 1000 nodes, averaged over 100 simulations.

The other networks seems to perform similarly, but lets explore it in more detail.



Number of different words over time for different Small World networks.

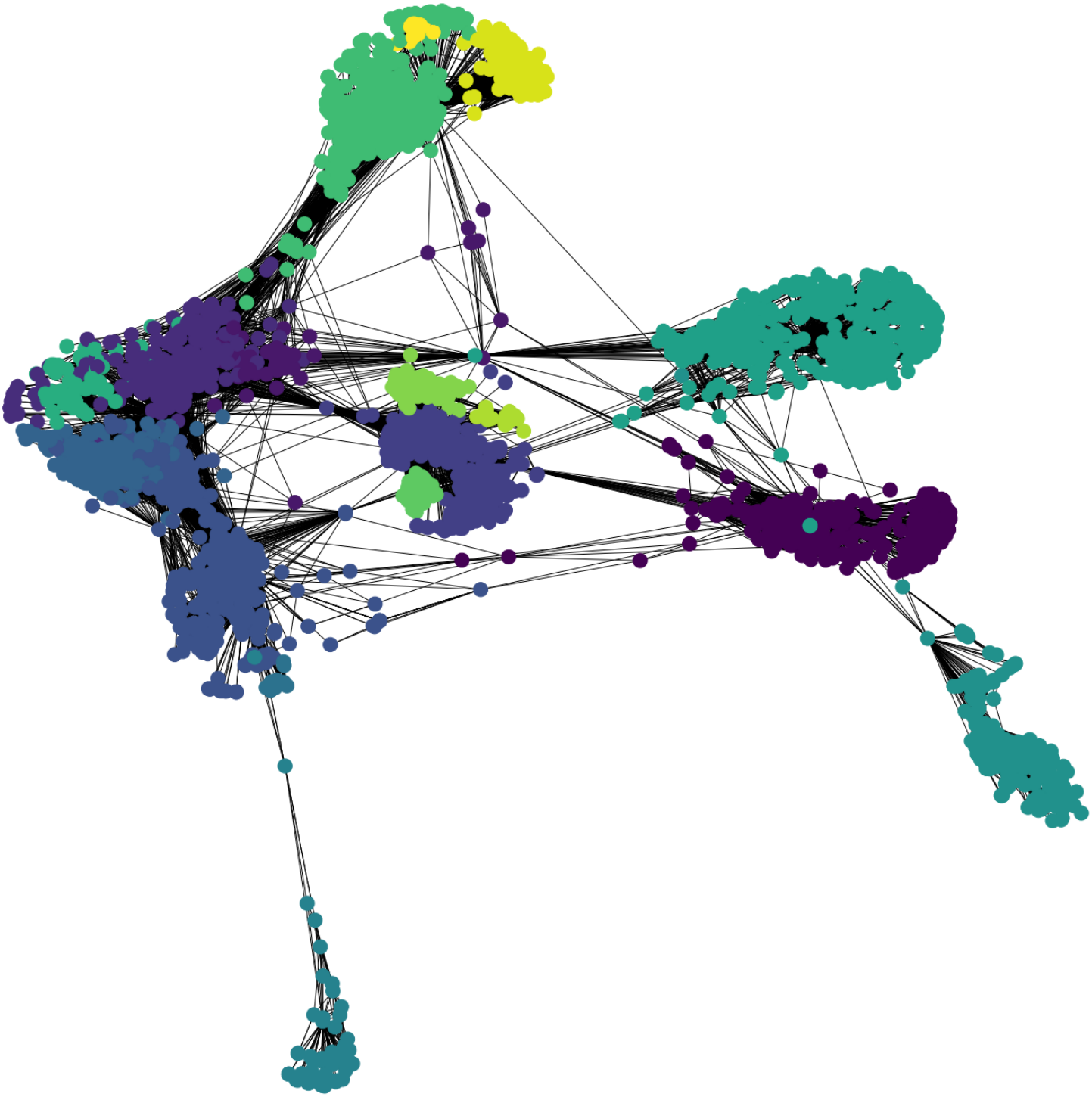
communication to be more efficient). It is interesting here to remark that the number of different words changes in time follows a power law, and there is a sudden cut of (in the image, we see it in the line of p=0.08). As we increase the rewiring between our agents, so more distant nodes get in touch, although less neighboring nodes communicate, the consensus is achieved earlier, so it is more efficient. So far this seems pretty intuitive. We can see what happens in the scale-free network.



Evolution of the total words in a Barabási-Albert network of 1000 nodes, over time, in thedirect and reversed game.

We can run the dynamics in an inverse way, so to pick up first the hearer and then select a speaker among its neighbors. In the previous networks we had a symmetry, but now the network is heterogeneous. That means that if we pick up first the hearer, the speaker we are going to select has a high probability of being a hub, so the hubs are going to act more actively as a speaker, therefore to spread its particular definition of the object more efficiently, and decreasing the consensus time, as we can see in the previous figure. If we do the direct game, the speaker has the same probability of being anyone, so everyone will play as a speaker on average the same number of times.

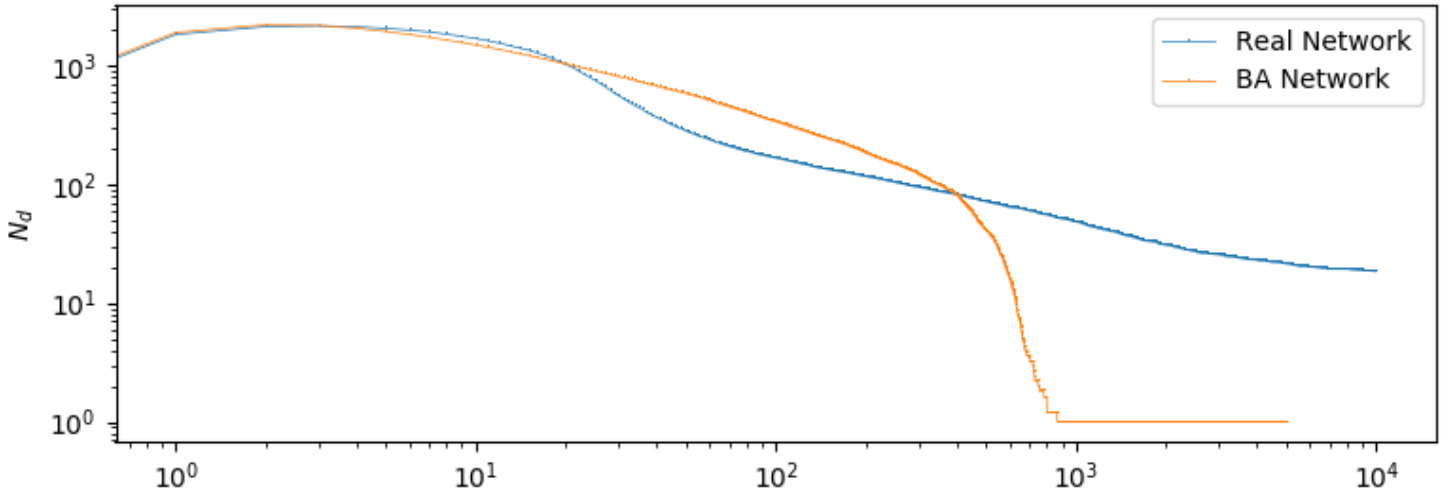
I just want to show you one last amazing result. In graph theory, there is a whole world of study in order to classify communities, for example with clustering techniques. It is important for detecting social or biological communities, and its presence make the topology more intricate, thus dynamics behave in more subtle ways. This is a hard problem that often needs lots of memory and computational time, when we have large real networks.



Communities of the Facebook network analyzed

I did the following: I took one list of neighbors of a Facebook social network, available here https://snap.stanford.edu/data/egonets-Facebook.html, and I performed community detection with a widely used algorithm called Louvain method. I obtained the 16 communities remarked in the left figure. Then, I performed the dynamics of the Naming Game in that network and, guess how much different words the system ends up to have! Not 16, but 17±2. Having in mind that there is not an absolute algorithm that detects perfectly communities (indeed the definition itself of communities is still a research problem), we can state that this dynamics can be used as a

would return some communities, the Naming Game does not. We see that the consensus is not absolute now for the network with real communities (blue line), since every community has its own object designation.



Number of different words in the real network, best approximated with a scale-free network, but with real communities.

Conclusion

With this example, although not all results and programs are shown in detail, I hope I manage to transmit some intuition about the power of the approach of creating an agent-based model to test some features and explore a wide range of properties. You can use the toy model and then change the topology, change details of the dynamics to do it more realistic, or whatever you can imagine, and see what useful information you can extract of it. You can check the whole programs I used and the figures took here: https://github.com/reymom/Naming-Game.