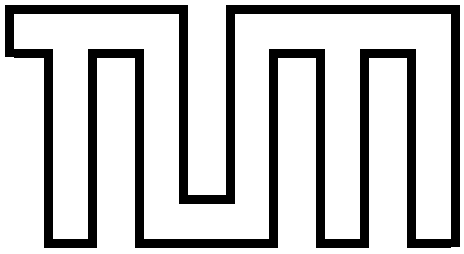


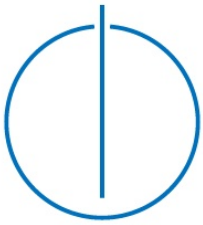
Technische Universität München

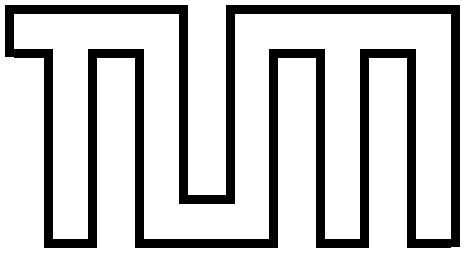
Fakultät für Informatik

Master's Thesis in Informatik

An Email-Centered Approach to Intelligent Task Management Using Crowdsourcing and Natural Language Processing

John Doe





Technische Universität München

Fakultät für Informatik

Master's Thesis in Informatik

An Email-Centered Approach to Intelligent Task Management Using Crowdsourcing and Natural Language Processing

Ein Email-basierter Ansatz für intelligente Aufgabenverwaltung mit Hilfe von Crowdsourcing und Natural Language Processing

Author: John Doe

Supervisor: Prof. Dr. Johann Schlichter

Advisor: Dr. Wolfgang Wörndl

Submission: DD.MM.YYYY

I assure the single handed composition of this master's thesis only supported by declared resources.

München, DD.MM.YYYY

(John Doe)

Abstract

English abstract.

Inhaltsangabe

Deutsches Abstract.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

1.2 Contributions

Chapter 2

Channelmodel

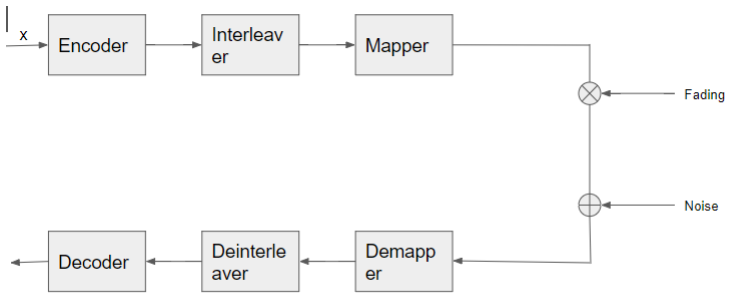


Figure 2.1: Channelmodel for general Transmitter/Receiver Chain

Above depicted is the channel model we will be working with in further simulations, which will include a Mapper/Demapper, Encoder/Decode, Interleaving, and the needed channel. We will briefly go over all blocks depicted in the graphic above.

2.1 Encoder/Decoder

For our Encoder/Decoder block we will be looking at a WiMax LDPC code according to the standard IEEE 802.16e. This standard code is used in small and medium distances in urban areas and fits our model quite well. LDPC, which stands for Low Density Parity Check, is a linear error correction code. While it has heavy computing demands, with the growth of computing power it sees more and more use in everyday use. In our case with WiMax we have different given blocksizes ranging from 576 codewords upto 2304. The rates given are 1/2, 2/3, 3/4, and 5/6. We also only look at coding class A for our simulations.

2.2 Bitinterleave/Deinterleaver

With the help of bitinterleaving we can avoid any kind of "bursterrors", i.e. we avoid any longer blockerrors. The interleaving in MATLAB is accomplished by creating a random permutation array k. The permutation array can be used to randomly shuffle our codeword and also return them back to default at receiver side. In our simulations we will create a codeword out of random bits. This codeword will be randomly shuffled again and reconstructed after demodulation has been done.

2.3 Mapper/Demapper

The mapper or modulation is used to assign a specific codelength a symbol which is transmitted. The symbols are located in a real/imaginary plane, also called Inphase/Quadrature Planes (I/Q). With the distance from the nullpoint of the axis giving us the amplitude of our signal and the angle to the real axis the phase shift. There are many forms of modulation schemes, with the most common ones being M-PSK, M-FSK, M-AM and M-QAM. For our simulations we will have a further look in QPSK, 16-QAM and 64-QAM, which are depicted below.

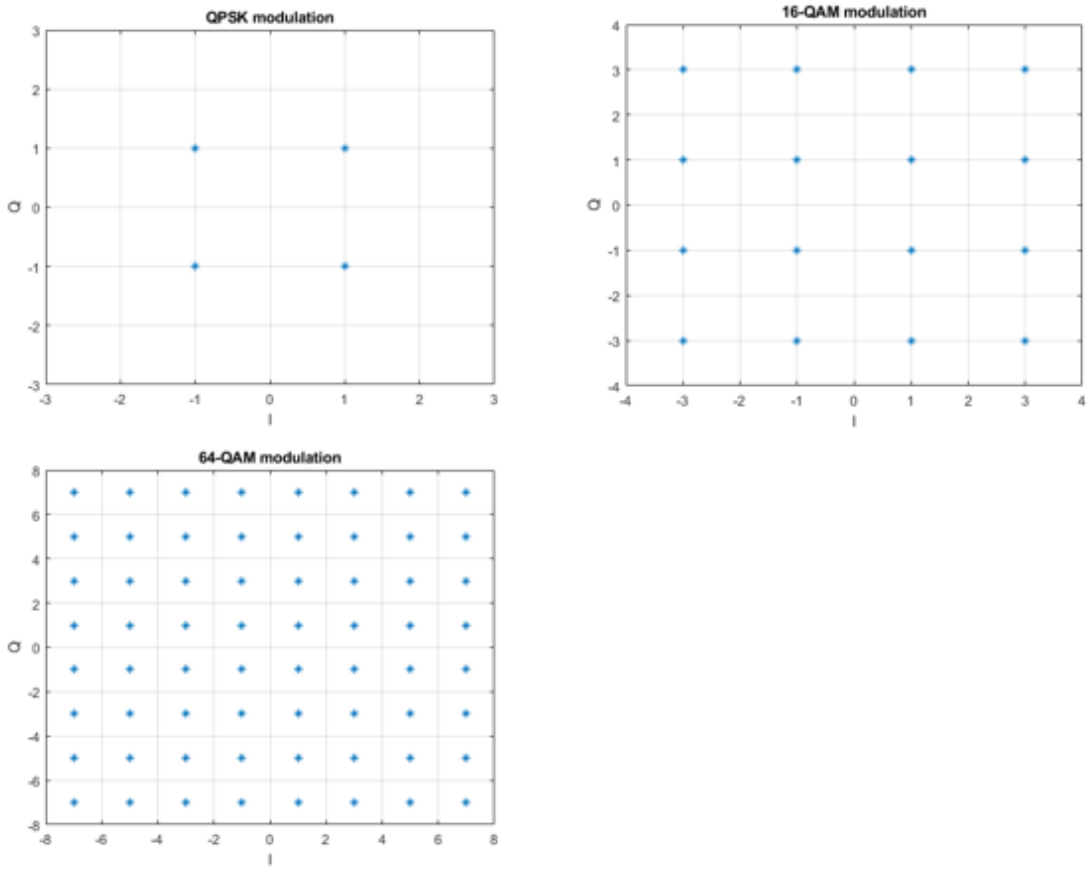


Figure 2.2: Modulation in I/Q planes for QPSK, 16-QAM and 64-QAM

QPSK, which stands for Quadrature Phase shift keying is one of the easier models we will be looking at. The symbols all share the same amplitude and only differ in their respective phase angle. With the information entropy $S = log_2(M)$ we can identify the maximum number of bits we can assign in every symbol, with M being the number of symbols in the modulation scheme. So for QPSK the number of bits per symbol amounts to 2.

With M-QAM, Quadrature Amplitude Modulation, we add the phase shift already implemented into QPSK with the additional differentiation with the amplitude of symbols. For QAM we send signals which differ in their phase shift and also the amplitude. For 16-QAM we get a maximum of 3 bits per symbols and for 64-QAM 4 bits per symbol. The modulation schemes makes it possible to increase our rate of transmission and is used for any kind of practical transmission of information.

2.4 Channel

The channel can be modified in many different ways. We can apply different sources of noise or fading, which can relate to realworld interferences. Some interferences experienced in real life transmission are e.g. thermal noise, distance, doppler effect and reflection of signals. To approach those kind of interferences there are many different channel models in simulations, like an AWGN-Channel or Rayleigh/Rician fading. We will have a further look into the AWGN-Channel and the Rayleigh fading.

2.4.1 AWGN-Channel

Lookup math The easiest channel manipulation is to add random gaussian noise to the channel, also commonly known as AWGN-Channel (Additive White Gaussian Noise). Like the name says we will add noise which is randomly distributed in a gaussian distribution. The probability density function is defined as follows:

$$f(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} * e^{-\frac{(x-\mu)^2}{2\mu}}$$

With x being the aquired point, μ being the mean or expection of the distribution and σ^2 the variance of the distribution. For our gaussian noise we will take a mean of 0 and a variance of 1, which will simplify further calculations in the following chapters. We will also always look at complex gaussian noise in our simulations. More or less every communication link will have some kind of gaussian noise interference, so we will add the AWGN-Channel to every simulation we run.

Add picture of AWGN, pdf or distribution

2.4.2 Rayleigh-Channel

Lookup math Another common channel model used in communication theory is Rayleigh fading. Rayleigh fading is used to simulate multipath reception, which means that for a

receiver antenna in a wireless link there are many reflected and scattered signals reaching it. This results into construction or destruction of waves. Rayleigh distribution can be defined like this: $R = \sqrt{X^2 + Y^2}$ with X and Y being two independent gaussian distributed random variables. Further calculations will lead to the following pdf:

$$f(x;\sigma) = \frac{1}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$

Add pictures, spreading, reflection...

Chapter 3

Capacity in an AWGN channel

We will now look into the maximum capacity we can achieve for our communication model in Chapter 2 with added AWGN noise.

3.1 Capacity and Monte-Carlo-Simulation

In general capacity C can be defined as the rate R at which information can be reliably transmitted over a channel, which means as long $R \leq$ C we can achieve a transmission without errors even with noise. All the capacities we wil be looking at will be for complex channel models.

For a AWGN-Channel we will have a simple channel model defined by Y = X + N with X $\sim N(0,sigma^2)$ and N $\sim N(0,1)$. With this our received signal Y will have a distribution of Y $\sim N(0,sigma^2+1)$ under the condition that X and N being independently distributed. We will calculate the capacity as the maximum of mutual information I between X and Y:

$$C = max(I(X;Y)) \tag{3.1}$$

with X and Y being to independent randomly normal distributed variables. With the maximum mutual information we calculate the maximum information we can achieve with the given parameters, like modulation, encoding, channel.
For the mutual information we can further part it into the differential entropy:

$$I(X;Y) = h(Y) - h(Y|X) \tag{3.2}$$

With differential entropy being defined as:

$$h(Y) = \int p(y) * [-log(p(y)]\mathrm{d}x \tag{3.3}$$

We will now apply the simulation of monte carlo to turn our integral into an addition. The Monte-Carlo-Simulation will be further explained in the following chapters.

$$h(Y) = h(X+N) = log(\pi * e^{\sigma^2+1}) \quad \text{and} \quad h(Y|X) = h(N) = log(\pi * e^1) \tag{3.4}$$

Further calculations will lead us to the final equation for the capacity in an AWGN-channel:

$$C = log(1 + \frac{\sigma^2}{N}) \tag{3.5}$$

With this approach we have good approximation values for further calculations with added modulation schemes. It is given that for only AWGN the capacity is at his maximum, there should be no capacity value over the calculated ones here.

3.1.1 Approach in Matlab

The above mentioned formula 2.4 will be simply implemented in MATLAB. With our noise being randomly distributed around 1 our formula simplifies even more into:

$$C_{\text{AWGN}} = log(1 + SNR) \tag{3.6}$$

The SNR here must be transformed into power and not in decibel.

3.1.2 Monte-Carlo-Simulation

Monte Carlo Simulation is widely used in stochastic to get solutions for random experiments. It is used to solve analytical unsolvable problems numerically. MC is based upon the law of large numbers, which says that a large number of performing the same experiment will lead the average of the results close to the expected value. We take this as our bases to get reliable results. The Monte Carlo simulations will be used for two calculations, once already used above for calculating the differential entropy and later once to calculate our theoetical Rayleigh fading curve out of AWGN.

3.2 Capacity for QPSK and M-QAM

Now we will look into different modulation schemes, which were already mentioned in Chapter section 2.3. We will implement these modulation schemes into our capacity calculations in an AWGN channel.

3.2.1 QPSK

For QPSK we will have 4 symbols and resulting 2 bits per symbol. Before any simulation or calculation were run we can already be sure that we will not pass the upper bound of 2 bits/Symbol. So the plot will approach the 2 bits/Symbol for high SNR. After creating a random codeword modulated with the fitting modulation scheme. Noise is added to the signal, which is than received as the bit array Y. The next step to calculate the capacity differs from above.
We know that our signal is normal random distributed variables and we have to calculate the differential entropy for h(Y), which is:

$$h(Y) = \sum_{n=0}^{N} (-log(p(y_{\mathrm{n}}))) * \frac{1}{N} \tag{3.7}$$

with p(y) being the probability of y for a normal distributed variable and N the codelength.

$$p(y) = \frac{1}{n * \pi} * \sum_{i=1}^{n} (e^{y - x_{\mathrm{i}}}) \tag{3.8}$$

Here we only need to watch out for the number of symbols in the modulation scheme. For QPSK we have a n = 4, 16-QAM n = 16 and 64-QAM n = 64.
For the QAM modulation only the above mentioned parameter n must be changed.

3.2.2 Results

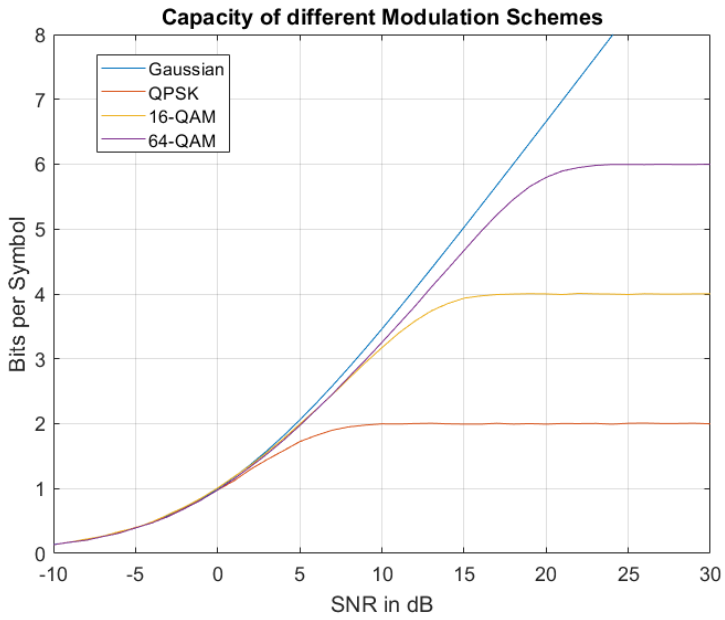


Figure 3.1: Capacity plot for general AWGN-channel, QPSK, 16-QAM and 64-QAM

The results of the calculation in MATLAB can be seen above. We can clearly see the modulated channels approach the desired bit/symbol in a good SNR to bit/symbol rate. The gaussian channel clearly outperforms the modulated channels, clearly seen after 0 dB SNR.

Compare with book capactiy!

Chapter 4

Transmitter Receiver Chain in MATLAB

We will now focus in creating a functioning Transmitter-Receiver chain to simulate a wireless communication as real as possible. The blocks for the communication link were shortly introduced in the beginning, but will be explained further in the following chapters. With LDPC WiMax we use a common communication protocol, which simulates a real channel quite well. Furthermore we will use soft mapping to reconstruct our symbols not hard decoding. Later on I will explain my reasoning behind it.

4.1 LDPC and the CML Library

With a given codeword x of length n and a generator matrix $G = [I^T|P]$. The parity check matrix texitH can now be derived as $H = [-P^T|I_{\text{n-k}}]$. With the parity check matrix H and a code $C = xG$ the condition for $cH^t = 0$ must be fulfilled for the codeword to be valid. Also with a parity check matrix error correction can be done, that means for single errors we get in our codeword the parity check matrix can selfcorrect our code. This whole process in MATLAB can be computed with the help of the Coded Modulation Library (CML). For this we have the given function "*InitializeWiMaxLDPC*" to create the parity-check.matrix, "*LdpcEncode*" and "*MpDecode*" to encode and decode our codeword. We decided on a length of 2304, the maximum length that can be send, and self correcting for 50 iterations to be sure to correct as many errors as possible that we receive at the end of our communication chain.

4.2 Soft-demapping vs. Hard-demapping

These two approaches will result in rather different result in any kind of simulation. We will have a look in both approaches and will compare their unique advantages and disadvantages. For hard-demapping a received symbol is compared to a given fixed threshhold. At every sampling instant the receiver will decide the state of the bit, either "0" or "1". Hard-demapping uses the minumum Hamming distance to make a decision, which means that bitrow from our receiver is compared to every available constellation point. For every bitdifference between bitrow and constellation point will add to the Hamming distance. In the end the receiver will make a decision by taking the constellation symbol which compared to the created bitrow resembles the most, that means the one with the lowest Hamming distance.
Major difference to hard-demapping the soft-demapping will use the euclidean distance to make a decision. It will use additional informations supplied by us to make a decision. While hard demapping has no info about the reliability of the receivers decision, soft demapping will gives us exactly this. With the eucladian distance we calculate the distance between received symbol to every constellation point. Furthermore we will use the loglikelihood ratio to calculate the reliability with the euclidian distance. While hard-demapping is fast and easy to implement in a system it gives us no reliability and as good of performance as soft demapping. In the end it is a decision based on a balance of computing complexity and performance gain. With many modern systems achieving great computing capability and our desire to create a channel as good as possible we will decide to use soft-demapping. In the next section we will have a further look into soft-demapping and LLR based on a QPSK example.

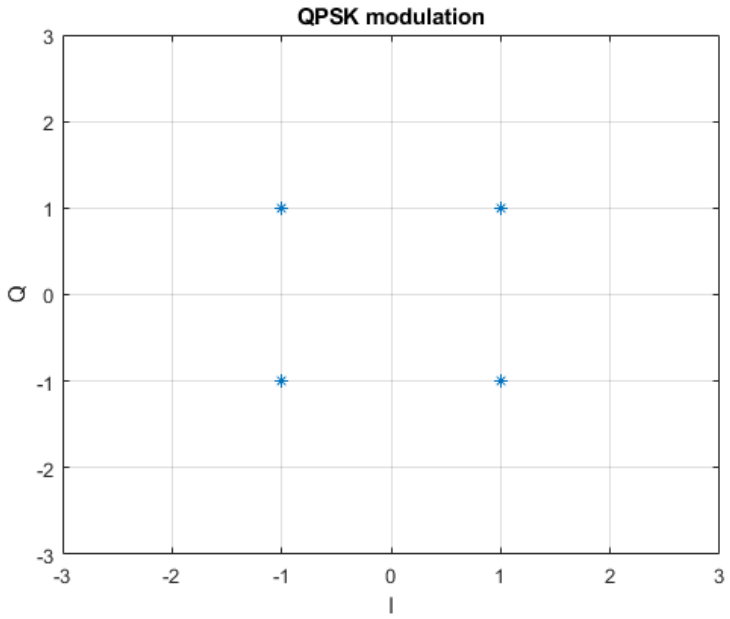


Figure 4.1: Capacity plot for general AWGN-channel, QPSK, 16-QAM and 64-QAM

With QPSK we have 4 different symbol constellation also depicted above: (0,0),(0,1),(1,1) and (1,0). The loglikelihood ratio is defined as below

$$L^n = log\frac{P(Y|B_1 = 0)}{P(Y|B_1 = 1)} = log\frac{P(Y|X_1) + P(Y|X_2)}{P(Y|X_3) + P(Y|X_4)} \tag{4.1}$$

4.2.1 Results

After receiving our demodulated symbols we can compare those to our codeword we initially send. With this we will determine the frame errors we got for the whole transmission. A frame is defined as a whole codeword length, that means for us it is 2304 bits sent. We have to simulate at least 100000 of those codewords to receive a reliable error rate. Our error rate $= \frac{\text{frame errors}}{\text{number of frames sent}}$. With 100 errors being a reliable number we can also prematurely interrupt our simulation after 100 errors to save simulation time.

4.3 FER and comparison with capacity plots

Add FER points with respective capacity plots

Chapter 5

Communication link for Rayleigh fading channels

Mention slowfading We will do the same as before but also add the fading coefficient H to our channel. The fading coefficient is represented by rayleigh fading, which was introduced in chapter 2.1... For fading our received signal changes in this way:

$$Y = \sqrt{\sigma^2} * H * X + N \tag{5.1}$$

With the fading being unknown to our receiver we need a way to extract or estimate the fading coefficient in the channel. An efficient and easy approach to this it to insert a pilot symbol X_p before the transmission. We also will divide our whole codeword in single blocks T which will range for blocksize equal to one symbol up to the whole codeword being one codeword. For every block we will insert one pilot symbol at the beginning.

Graphic for block + pilot

Our pilot symbol will have the default value of 1, which is also known at the receiver side. This means to estimate the fading we will do this:

$$Y_p = \sqrt{\sigma^2} * H * X_p + N \tag{5.2}$$

which leads to:

$$H_{\text{est}} = \frac{Y_p + N}{\sqrt{\sigma^2} * X_p} \tag{5.3}$$

With this we get a proper estimation for the fading coefficient, but its estimation is highly dependable of the strength of fading and SNR. With higher SNR we receive better estimation not disturbed by the noise as much. And with lesser fading, close to 1, we do not receive a weakened signal, which is hard to distinguish from the noise.

Maybe add graphics which shows the single scenarios

With the estimated fading coefficient the symbols can be reconstructed.

$$Y_{\text{est}} = \frac{H}{H_{\text{est}}} * \sqrt{\sigma^2} * X + \frac{N}{H_{\text{est}}} \tag{5.4}$$

and with H_{est} being close to H we get

$$Y_{\text{est}} = \sqrt{\sigma^2} * X + \frac{N}{H_{\text{est}}} \tag{5.5}$$

which can be used to calculate the log likelihood ratio.

Maybe add the scatterplot with and w/ rayleigh fading

5.1 Theoratical rayleigh fading FER constructed out of AWGN-Channel

For a proper simulation we will need a reference to compare our simulation results to. For this we will construct a theoratical FER plot for rayleigh fading out of the AWGN-channel. First step is the simulate the AWGN channel with the desired codelength over our SNR. In the previous chapters we have already proven that our simulations are match the theoratical curves. With the simulation for AWGN finished we can now start creating many random value (here n = 10000) rayleigh fading coeffiecients. For every single step of SNR, one step being one SNR, we will compute the SNR after rayleigh fading is created, that means $\text{SNR}_{=}$ SNR * H. The SNR for the fading has a corresponding FER-value which we will be add up and divide by the number of fading coefficients.

$$FER - with - SNRi = \sum_{k=0}^{n} FER(SNRi * H(k)) \tag{5.6}$$

Add plot from AWGN

It can seen that for the AWGN channel we will reach the error floor really fast at around 3 SNR. For any snr-value which was not simulated for, here only for 0 - 5 SNR, we will add virtual value. While the frame error rate of the AWGN channel reaches a minumum value for high SNR it will never reach 0. In our case we will calculate the theoratical rayleigh FER with error floor 0 and once with error floor 10^{-6}, just to show the drastic difference in performance with different error floors. Later on we will also prove that the assumend error floor of 10^{-6} comes close to the real error floor.

Plot for FER with error floor 0 and 10^{-6}

5.2 Rayleigh fading FER with AWGN channel

Add different plots 1. Simulation with perfect channel knowledge 2. Simulation with estimated coefficient 3. Different blocksizes T=N/2, T=N/16 Explain difference and why? We can clearly see a distinct performance difference between the two error floors. OH WOW! SURPRISE!

5.3 Increase power of pilot symbol

5.4 Increase of pilot symbols in one block

5.5 Results and comparison with AWGN channel

Chapter 6

Conclusion

6.1 Comparison between fading and AWGN channel

6.2 Fazit