Assignment 4: Natural Language Processing

Johan Östling - 25h
Hanna Olsson - 25h

February 14, 2023

1 Reading and Reflection

1.1 Hanna

Speech recognition is another AI problem that has a very wide range of approaches. This field deals with teaching machines to transcribe spoken language into text. Approaches here range from traditional models like Hidden Markov Models (HMMs) to more recent deep learning-based models like recurrent neural networks (RNNs) and attention-based models.

One similarity between rule-based translation systems and state-of-the-art neural systems is that they both use pre-existing resources. They both use pre-existing resources such as bilingual dictionaries to enhance their performance.

For domain-specific translation, the old school solution might be preferable. In some specialized fields, such as medicine, law, or technology, a rule-based system might be favored due to its ability to be customized to the particular language and vocabulary used in that field. When used in these cases, a rule-based system can yield results that are more precise and uniform compared to a neural or statistical system that has been trained with a broader range of data.

1.2 Johan

Yes, I can think of computer vision, which is the development of machines that can interpret visual information. In this area, computer vision algorithms started based on rule-based systems to detect edges and identify patents. Then machine learning algorithms were used, such as Bayesian methods, neural networks and decision trees. Today, deep learning has been introduced to this area and computer vision has improved a lot over the years since its beginning.

They are both similar in the way that they require bilingual dictionaries to be able to perform better. Both system also aims to automatically translate a language to another language.

I would think that in a certain application, a rule-based system would perform better. For example, in translating medical drugs. Then a rule-based system would be preferable since it does not use probability to determine the translation, and since this application only needs to translate a very specific word or phrase, a rule-based seems like a good option.

2 Implementation

2.1 a)

We choose Swedish and English. The 10 most frequent words in these languages were:

TOP 10 ENGLISH

	word	count
0	the	19322
1	of	9312
2	to	8801
3	and	6946
4	in	6090
5	is	4400
6	that	4357
7	a	4269
8	we	3223
9	this	3222

Figure 1: Most Common English Words.

TOP 10 SWEDISH

	word	count
0	att	9181
1	och	7038
2	i	5951
3	det	5687
4	fr	5274
5	som	5028
6	r	4124
7	av	4013
8	en	3724
9	vi	3211

Figure 2: Most Common Swedish Words.

For the probability of the word *Zebra* we got 0 %, and of the word *Speaker* we got 0.003915304140 %

2.2 b)

When we try to compute the sentence 'How are you' which consists of words that definitely exist in the data, we get the probability of 1.8269817804241945e-06.

If we try a sentence that contains a word that does not exist in the data, *zebra* for example, the sentence 'I am a zebra' gets the probability of 0. This is because the probability of zebra succeeding a word in our data set never happens. So 'zebra' never follows after 'a'.

For a very long sentence, we asked ChatGpt to write a 100 words sentence, using common English words. It provided this sentence:

'Despite the fact that the world is currently facing numerous challenges, including political upheaval, economic uncertainty, and the ongoing global pandemic, people from all walks of life continue to persevere and find ways to adapt, often through innovation and collaboration with others, as they strive to create a brighter future for themselves and their communities, demonstrating the resilience and ingenuity that is at the core of the human experience, and serving as a reminder that even in the face of adversity, there is always hope for a better tomorrow'.

This sentence gave the probability of 0. This is because our model computes the portability of a sentence as the product of the conditional probabilities of succeeding word which will go to zero when doing many multiplications. A possible solution would be to use log probabilities since you then get to add the probabilities instead of multiplying them, however we did not implement such a solution.

2.3 c)

For the translation model we followed the exact steps of the provides pseudo code. Our model have these lists of 10 words 'European' should most likely be translated to, for the first 10 iterations:

Iteration 1:
Top 10 words that 'european' is most likely to be translated to:
lugnas: 0.14285714285714285
flygsäkerheten: 0.11111111111111112
kriteriet: 0.10000000000000003
kärnpunkt: 0.10000000000000002
skadat: 0.1
hörd: 0.09999999999999999
straffbestämmelser: 0.09999999999999996
aktioner: 0.09375000000000001
importeras: 0.09090909090909094
attraktivt: 0.09090909090909093

Iteration 2:
Top 10 words that 'european' is most likely to be translated to:
europeisk: 0.22491759450179935
europeiska: 0.19678417831278017
lugnas: 0.166922653072531
europaparlamentet: 0.1556252817946196
europeiskt: 0.1348151646145679
kriteriet: 0.11020530020087854
hörd: 0.10802553587813651
csu: 0.10517039509399224
flygsäkerheten: 0.10383661944662086
europaparlamentets: 0.10176905023207067

Iteration 3:
Top 10 words that 'european' is most likely to be translated to:
europeisk: 0.5222941262545281
europeiska: 0.44523178921904927
europeiskt: 0.34834284592855547
europaparlamentet: 0.30168221064672984
lugnas: 0.22134224799891608
europaparlamentets: 0.2203019099491861
valdeltagandet: 0.13214902788095412
sa: 0.13120319681760023
csu: 0.12917703314153575
unionen: 0.123112427497419

Iteration 4:
Top 10 words that 'european' is most likely to be translated to:
europeisk: 0.7041392308052049
europeiska: 0.6243513521748112
europeiskt: 0.5464352026952202
europaparlamentet: 0.3924849970924109
europaparlamentets: 0.313651871039285
lugnas: 0.2708122876510805
bitter: 0.19630153093306724
valdeltagandet: 0.19355925868839158
sa: 0.18218625603507732
europaval: 0.16027226312537216

Iteration 5:
Top 10 words that 'european' is most likely to be translated to:
europeisk: 0.8006476792230388
europeiska: 0.7311339324880528
europeiskt: 0.6693649265378344
europaparlamentet: 0.4405999679894263
europaparlamentets: 0.3734124533797027
bitter: 0.3107153201202707
lugnas: 0.29600240934030614
valdeltagandet: 0.23323884130019254
europaval: 0.2275577049581496
sa: 0.21922974749904242

Iteration 6:
Top 10 words that 'european' is most likely to be translated to:
europeisk: 0.8561305646824401
europeiska: 0.794761233670848
europeiskt: 0.7438842526497179
europaparlamentet: 0.464391472847648
europaparlamentets: 0.4080443962328672
bitter: 0.40195952376559274
lugnas: 0.3059729219837553
europaval: 0.27954580363175946
europeiske: 0.275878127707056
valdeltagandet: 0.24610757737865563

Iteration 7:
Top 10 words that 'european' is most likely to be translated to:
europeisk: 0.8910497083851826
europeiska: 0.8341050381733579
europeiskt: 0.7914211440330429
europaparlamentet: 0.4765294950906772
bitter: 0.46722174376499287
europaparlamentets: 0.4280795964529708
europeiske: 0.31735503952750466
europaval: 0.3167612117542739

lugnas: 0.3090613572160085
jean: 0.27875252237078585

Iteration 8:
Top 10 words that 'european' is most likely to be translated to:
europeisk: 0.9146271575916656
europeiska: 0.8594324123027465
europeiskt: 0.8230080400032472
bitter: 0.5118797741232062
europaparlamentet: 0.48304655591591805
europaparlamentets: 0.44036944770615405
europeiske: 0.344934945539756
europaval: 0.34348861650874274
jean: 0.3145740691736398
lugnas: 0.30934154988821017

Iteration 9:
Top 10 words that 'european' is most likely to be translated to:
europeisk: 0.9314015074956385
europeiska: 0.8763816444060692
europeiskt: 0.8445636110349228
bitter: 0.5418902382077022
europaparlamentet: 0.48671229592559084
europaparlamentets: 0.44835094799560715
europeiske: 0.36289236645446077
europaval: 0.36249936522875514
jean: 0.34344102761701173
lugnas: 0.3086529848851774

Iteration 10:
Top 10 words that 'european' is most likely to be translated to:
europeisk: 0.943803601088794
europeiska: 0.8881608752975607
europeiskt: 0.85970964001552
bitter: 0.5624575722777314
europaparlamentet: 0.48889039420563213
europaparlamentets: 0.45375300882986486
europaval: 0.37595962220000306
europeiske: 0.3742042386781344
jean: 0.3664752238715147
lugnas: 0.307792431022809

As seen the first iteration seems very random and 'europeisk' which is the correct translation was not even on the top 10. But for the 10th iteration 'europeisk' is top word, followed by very similar words. This indicates that our IBM model works as it should.

2.4 d)

To make the decoder we combined the language model and the translation model. For each word in the sentence, we extracted the top 5 most likely translations based on the translations probabilities in t. Then we created a list with possible translations for the sentence by taking the combination of all the translations for the words in the sentence. Then we iterated through this list and computed the probability for each sentence using the language model. The method finally returns the translated sentence with the highest probability.

When translating the sentence "han är bra" we get "he is fine" and for "varför är detta bra" we get "why is this good" which are very good translations for such a simple model in our opinion. However, for sentences where the flow is different in Swedish compared to English, such as "Vad pratar ni om", the quality of the translations decreases (in this case we got "what misappropriation you if"). Another problem area are sentences that contain words that the list of source words does not contain. Our approach for these words was to just pick five random words as the top five translations to make the code run smoothly, but of course, this is not ideal since it will basically produce nonsense translations. A better way would be to look at a word in the source words that are the most similar and then find the translations for this word instead. So if the Swedish sentence was "han spelar fotboll" and "fotboll" does not exist in the Swedish training set we should look for words similar to "fotboll" and might find that "fotbollar" exists and then translate that word instead. This would probably result in a translation like "he plays footballs" and even though it is not correct it is still understandable.

We think that the ideal model would consider all possible translations of a word and look at all possible combinations of these. However, to make the method run quicker we chose to only look at the top 5 most likely words. This is still very demanding and especially for long sentences. For example our decoder took 37 seconds to run for "han är bra" and 4 minutes to run for "varför är detta bra". The run time increases exponentially with the number of words in the sentence since the number of possible translations for a sentence is $5^{\#words}$. If we were too look at all possible translations this would simply be computationally infeasible with our implementation. It probably could be done more efficient by using other data structures and vectorization n instead of for loops.

We also thought about further ways to improve the performance of our model. Now we look at the probability of the sentence by looking at all the combinations of words and choosing the order with the highest probability. This could be improved by implementing constraints, such as the first and last word when translating from Swedish to English, never would be switched in the translation. This removes some sentences that our model would need to go through. There could exist multiple constraints to limit the possible combinations in the translated sentence, so if we further wanted to improve our model, we would dive deeper into this. However, this requires human competence about language which makes the process more time consuming.

3 Discussion

3.1 a)

For manual evaluation of a translation system one or multiple humans can grade the translations based on different factors such as accuracy, fluency and cultural appropriateness. Since the translation model wants to mimic a system that humans have created (language) an obvious advantage of this is that a human is arguably better at judging how similar a translation is to what humans would produce. This is because humans can take into account small nuances in the language which are very hard for a machine to learn. However, this approach is very time consuming and it is hard to standardize the evaluation since different evaluators have different views on what is correct.

For automatic evaluation a machine could simply check the n-gram overlap between the translated sentence and the matching human translation in a test set. If the overlap is high, the model will get a high score. This method is fast and scalable and suitable for large datasets and remains objective in its evaluation. The disadvantages are that it is hard to capture nuances in language and it might not take fluency and cultural appropriateness into account either.

For a translation system to be considered "good" it should accurately convey the meaning of the source text, be fluent and culturally appropriate. As we have seen, the two different approaches are good at maximizing different aspects of a "good" translation which might indicate that a mix between the two is optimal for evaluating how good a translation is.

3.2 b)

We believe that the model was trained on biased data where females and males were not equally associated with different professions. This means that the data probably had more sentences where males were associated with programmers and females with nurses since it is quite common, in the real world, that males are programmers and females are nurses. This really reflects the stereotypes of society and we would not consider it a feature since it can lead to unfair and incorrect predictions. Even though a person translating "Ta on arts" in most cases might be talking about a man, this translation model will almost always perform incorrectly when the person is actually talking about a female. To fix this issue a more balanced training set could be used where both genders are represented equally in relation to different professions. We do however realize that this can be hard since training data often comes from already existing sources (where these stereotypical patterns exist) and that producing additional training data manually is very time consuming.

3.3 c)

Since the model learns from training data that is most commonly taken from the real world (like subtitles and bilingual transcriptions) it has learned that bat, when associated with words such as "ball" and "hit", is much more likely to be a "club" than a mammal. And when associated with "insects" it's more likely that it is a mammal. However, for the third sentence, the model might have encountered about the same number of occurrences of the two definitions of bat in relation to "forest" and "lives". It could also be the case that the model has seldom encountered the word bat in this context. This makes the model uncertain and it comes up with a hybrid word. So in conclusion the model uses context to dis-ambiguous the word "bat" and for the first two cases the context made it clear what was meant but in the third case the context was not enough for the model to make the right translation. We also thought that the model that translated the third sentence has learned that when translating from English to Swedish, it is sometimes necessary to combine words. Hence it merged words in a "weird" manner to create "fladderträet" to maximize the likelihood of a correct translation.

4 Summary and Reflection

4.1 Hanna

4.1.1 Natural language processing lecture 7/2

Language processing is a big field within AI research and applications. The applications of NLP range from simple to hard, with spam filtering being regarded as simple and machine translation being more complex. Language differs from other types of data in many aspects since it is discrete, structured, diverse and sparse. These qualities make it harder to work than other data. The tasks for NLP can be divided into categorization, tagging, parsing and generation. Before the 80's rule-based methods dominated NLP but after this point more statistical and machine learning models were developed, and today these models completely dominate NLP. A common method for NLP was to divide the task into subtasks and solve the problem in many steps but today it is much more common to let a complex ML model solve the task directly. This is beneficial since it minimizes the possible errors, but these models can also be very messy to work with. They have many hyper parameters, convergence problems and they also consume a lot of energy.

In machine translation the goal is to generate a text in a target language given a text in a source language. This process tries to mimic the human translation of words, which is that you hear a word and then get a mental picture which you translate to the new language. To implement this an interlingua is needed, which is some data structure that is independent of language. However, it is very hard to find such an intermediate representation that is powerful and expressive when it comes to general language, but for some specific cases this method can be used. There are also data-driven machine translations systems which can be either statistical or neural.

There are five IBM models of increasing complexity that are used for word-based statistical MT. The goal is to develop a probabilistic model that allows us to translate by choosing the translation with the highest probability. Such a model can be divided into two parts: a language model and a translation model. A language model assigns a probability to a sequence of words and the translation models allows us to compute things like P(English sentence — French sentence).

4.1.2 Follow up 10/2

A problem with the data in assignment 3 is that there was a distributional shift. Meaning that the train data is very different from the test data. The data points have a different distribution in the test domain. If the distribution from the training data is $p(X, Y)$ and the distribution of the test data is $q(X, Y)$, then $p(X, Y) \neq q(X, Y)$. A common assumption that we make in machine learning is that p(X,Y) = q(X,Y). Instead of assuming that they are identical we can assume that the input given a label is the same, but that the distribution of labels has shifted. You could also assume that $p(X|Y) = q(X|Y)$ but $p(Y) \neq q(Y)$ which is called co-variate shift.

To do research in ML there are many tools but there are much fewer tools for production. This is because it is hard to standardize since different companies have very different needs. The code that ML algorithms write is very different from the code that us humans write. This makes software engineering different from machine learning since it lacks many qualities such as version control and modality.

To keep track of data versions you can simply copy the current data and save it as version. A drawback is that there might be a huge overlap between the data versions. So another way to deal with this is to instead maintain the data which can be done by storing the queries executed on the database. A tool for data versions is Data Version Control.

4.1.3 Reflection

Developing our own classifier was harder than I initially thought. We chose to also write our own KM-mean algorithm which took a lot of time since it has many steps where you need to consider what the optimal choice is. For example what distance to use when assigning a data point to a cluster and how to recalculate the cluster centers. It also became clear that maybe not all data is suitable to be clustered.

One thing that I really learned during the assignment was the importance of using different metrics to determine how good a model is. Before I have mostly used accuracy as a metric but for our model it was clear that this did not give the entire picture at all. That is why we also choose to use precision, but looking back F1-score would also be interesting to see. I also learned that one common assumption when working with ML is that the training data is distributed in the same way as the test data. This gave me a lot of insight into why ML-algorithms can fail since this assumption does not always hold.

4.2 Johan

4.2.1 Natural language processing lecture 7/2

Natural language processing (NLP) is central in AI research and applications, and in AI philosophy many arguments involve language understanding. Such as the Turing test. Some applications of NLP are spam filters, translation systems and grammar checkers. Language differs from other types of "data" because it is discrete, structured, diverse and sparse. This makes the language more challenging than "classical" data. Typical tasks of NLP include categorization, tagging, parsing and generation.

Historically, NLP started out with rule-based methods, then went into data-driven methods, to then machine learning models. But there is still lots of rule-based systems in the industry.

The goal of machine translation is to take in a source language and generate text in a target language. This has been one of the main goals of AI, going back to the 1940s. One way to think about machine translation is to think about humans translating words. For example by having a mental representation of a word to then translate it. So the idea is to map a source language sentence into some "meaning representation" or interlingua. Then convert the representation into the target language. The challenge of this is to make the representation expressive enough. So these types of systems are difficult to implement but interlinguas can work in restricted settings.

On the other hand, most machine translation systems are data-driven systems. Instead of writing rules machine translation systems are trained on example texts. There are both statistical and neural machine systems.

A fundamental idea in neural machine translations is encoder-decoder or sequence-to-sequence architecture. This means encoding information in the source sentence. And then decode, based on the encoding, to generate the target language.

The IBM models are models for word-based statistical machine translation. There are 5 IBM models, and we will focus on IBM model 1. The idea behind these models are:

Probabilistic machine translation, which selects the most probable target language sentence from the source language. This probability can be estimated by taking the maximum likelihood of the target language from a language model times a translation model. A language model assigns a probability to a sequence of words, For example, P(W) = P(please, open, the, door) → probability of the next word: P(wn—w1,...,wn-1) = P(door — please, open, the). This is approximated with the Markov assumption and the probabilities are estimated in a bi-gram model. The next part is the translation model, which assigns a probability of a sentence in the target language given a sentence in the source language.

4.2.2 Follow up 10/2

First grading notes and an overview of how the assignment went were introduced. Then distributional shift was explained. This is when the data is not distributed in the same way as the training data, which may result in poor classifiers. We often assume that the training and test samples have the same distribution, but this does not hold. Then we discussed in groups how different issues relevant to the assignment could be solved, such that over fitting could be solved with the weighting of samples.

Then we moved on to deployment and ML operations. In deployment, far fewer tools are used for production compared to research. Much like traditional software, AI software needs maintenance. Which we read about in the assigned paper for the AI tools assignment. Then we moved on to ML operations and looked at the architecture of how the workflow looks like. We also crossed part in the architecture which we had done previously, such as data transformation and cleaning, and model training.

AI as software primarily changes for two reasons: Changes to model pipeline and changes to training data. These are intimately connected.

4.2.3 Reflection

The last module was interesting but difficult. The difficulty lay in debugging our module to find ways to improve it. After completing the K-means clustering algorithm, we got a poor score in terms of accuracy. Then we tried to look through the algorithm to see if we did any mistakes in it. Eventually, we came to the conclusion that the data in itself was the problem. That the K-means clustering method was not a good way of labelling and predicting PM high.

I learned about how to structure an AI system in a reusable manner, which I know is very helpful for future projects. I have also learned to test the model and evaluate it using different types of measurement. For example to look at precision in complement to accuracy. Another takeaway I have from the previous module is to consider the distribution in the training set vs the test set because this can lead to inaccuracies in the model.