UPPSALA
UNIVERSITET

UPTEC IT 23036

Examensarbete 30 hp

Oktober 2023

Automatic Voice Trade Surveillance

Achieving Speech and Named Entity Recognition in Voice Trade Calls Using Language Model Interpolation and Named Entity Abstraction

Martin Sundberg & Mikael Ohlsson

Civilingenjörsprogrammet i informationsteknologi

UPPSALA
UNIVERSITET

Automatic Voice Trade Surveillance

Martin Sundberg & Mikael Ohlsson

Abstract

This master thesis explores the effectiveness of interpolating a larger generic speech recognition model with smaller domain-specific models to enable transcription of domain-specific conversations. The study uses a corpus within the financial domain collected from the web and processed by abstracting named entities such as financial instruments, numbers, as well as names of people and companies. By substituting each named entity with a tag representing the entity type in the domain-specific corpus, each named entity can be replaced during the hypothesis search by words added to the systems pronunciation dictionary. Thus making instruments and other domain-specific terms a matter of extension by configuration.

A proof-of-concept automatic speech recognition system with the ability to transcribe and extract named entities within the constantly changing domain of voice trading was created. The system achieved a 25.08 Word Error Rate and 0.9091 F_1-score using stochastic and neural net based language models. The best configuration proved to be a combination of both stochastic and neural net based domain-specific models interpolated with a generic model. This shows that even though the models were trained using the same corpus, different models learned different aspects of the material. The study was deemed successful by the authors as the Word Error Rate was improved by model interpolation and all but one named entities were found in the test recordings by all configurations. By adjusting the amount of influence the domain-specific models had against the generic model, the results improved the transcription accuracy at the cost of named entity recognition, and vice versa. Ultimately, the choice of configuration depends on the business case and the importance of named entity recognition versus accurate transcriptions.

Teknisk-naturvetenskapliga fakulteten

Uppsala universitet, Utgivningsort Uppsala/

Handledare: Fredrik Lydén

Ämnesgranskare: Ginevra Castellano

Examinator: Lars-Åke Nordén

UPPSALA
UNIVERSITET

Sammanfattning

Denna masteruppsats undersöker effektiviteten av att interpolera en större generisk taligenkänningsmodell med mindre domänspecifika modeller för att möjliggöra transkription av domänspecifika konversationer. Studien använder ett korpus inom den finansiella domänen insamlad från webben och bearbetad genom att abstrahera namngivna entiteter som finansiella instrument, siffror samt namn på personer och företag. Genom att ersätta varje namngiven entitet med en tagg som representerar entitetstypen i det domänspecifika korpuset, kan varje namngiven entitet ersättas under hypotessökningen med ord som lagts till i systemets ordlista. Detta gör instrument och andra domänspecifika termer endast till en fråga om konfiguration.

Ett proof-of-concept taligenkänningssystem med förmågan att transkribera och extrahera namngivna entiteter inom den ständigt föränderliga domänen för finansiell handel utvecklades som en del av studien. Systemet uppnådde 25,08 Word Error Rate och 0,9091 F_1-score med användning av språkmodeller baserade på stokastiska metoder och neurala nätverk. Den bästa sammansättningen av olika språkmodeller visade sig vara en kombination av både stokastiska och neurala nätverksbaserade domänspecifika modeller interpolerade med en generisk modell. Detta visar att även om modellerna tränades med samma korpus lärde sig olika modeller olika aspekter av materialet. Studien ansågs framgångsrik av författarna eftersom Word Error Rate förbättrades genom modellinterpolation och alla utom en namngiven entitet hittades i testinspelningarna av alla konfigurationer. Genom att justera vilken mängd inflytande som de domänspecifika modellerna hade mot den generiska modellen förbättrade resultaten transkriptionsnoggrannheten på bekostnad av entitetsidentifiering och vice versa. I slutändan beror valet av konfiguration på användingsområdet av en implementation, samt vikten av entitetsidentifiering kontra korrekta transkriptioner.

Teknisk-naturvetenskapliga fakulteten

Uppsala universitet, Utgivningsort Uppsala/

Handledare: Fredrik Lydén

Ämnesgranskare: Ginevra Castellano

Examinator: Lars-Åke Nordén

Contents

1 Introduction

In September 15, 2008, the investment bank Lehman Brothers collapsed as a result of the American sub-prime mortgage market. This lead to a full blown financial crisis reaching world wide. As a response, the American government signed the Dodd–Frank Wall Street Reform and Consumer Protection Act (DFA) in Juli 21, 2010 [1]. The DFA brought additional and stricter regulations concerning supervision of financial markets, affecting federal financial regulatory agencies and a large portion of other entities in the financial sector.

Furthermore, the European financial market complies to an extensive set of regulation commissioned by European Securities and Markets Authority (ESMA) [2]. Financial institutions must record and analyze trading of financial instruments performed in agreement with a broker over telephone calls on the institution's Voice Trading platforms to comply with ESMA and DFA regulations [3]. Voice Trading surveillance is thus relevant for any financial entity operating in a global context. However, making such analysis requires at least as many hours of manual labour as the extent of the voice trade recordings. It is therefore reasonable to utilise software solutions that provide automatic transcriptions for these recordings but is there an accurate and cost effective way to do so?

Is it possible to build a system that automatically transcribes the recorded speech of a voice trade and efficiently extract market specific entity information and if that is the case, how accurate can it become?

By using non-domain specific models we expect the results to be poor in terms of both Named Entity Recognition (NER) and transcription accuracy. We expect the same to be true by using only smaller specialized models. Is it possible to combine a general model with a domain specific model to provide cost effective and yet accurate entity detection and domain specific transcriptions? If so, what relation does the transcription accuracy have with the entity detection accuracy?

2 Background

Creating what henceforth will be called an Automatic Voice Trade Surveillance (AVTS) system is closely related to the issue of automatically transcribing and finding patterns in spoken language. Great advances has been made in the field of Automatic Speech Recognition (ASR) in recent years that together with the use of the Deep Neural Network (DNN) machine learning approach has lead to a significant increase of applications in products aimed at business as well as the consumer market.

2.1 Automatic Speech Recognition

A typical ASR system, illustrated in Fig. 1, consists of a few main components. Its purpose, to find the most likely transcription w^* of a sentence $w = (w_1, w_2, w_3, ..., w_n)$ communicated as a speech utterance u in audio wave format [4]. The input is transformed from time to frequency domain and typically enhanced by removing noise and distortions. The processed signal $\hat{u}$ is subjected to spectral analysis to extract a vector X of suitable acoustic features to feed the following Acoustic Model (AM) component. The AM models knowledge about acoustics and phonetics which integrated with X generates an AM score $p(X|w)$. Furthermore, in the search for a word sequence hypothesizing the uttered sentenced the AM score is combined with a score $P(w)$ from the Language Model (LM) component which models valid expressions of the language in question.

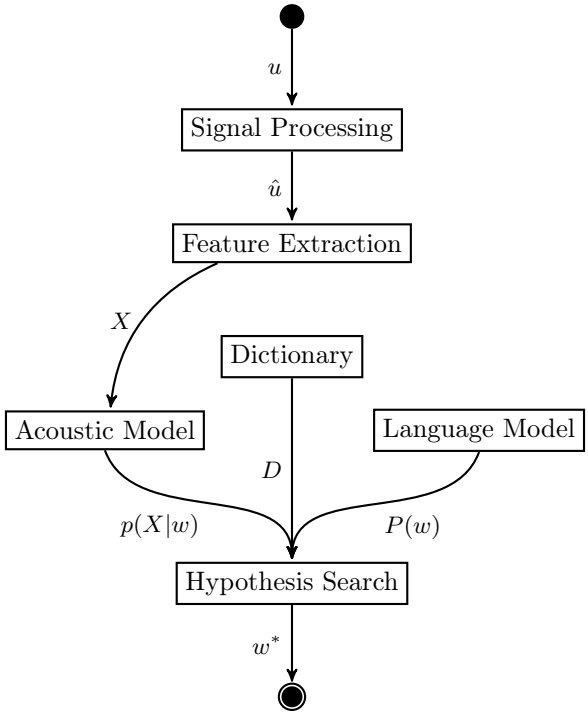

Figure 1: Typical ASR System Architecture [4]

2.2 Signal Processing & Feature Extraction

In order to decrease the Word Error Rate (WER), a measurement of the number of mistakes the system is making, the input signal, i.e. the recorded speech, can be subjected to a number of different techniques to reduce background

noise or increase the Signal-to-noise ratio (SNR). This is done in order to extract feature vectors that better represent the speech that was recorded. For example, the open-source library Sphinx 4 developed by Carnegie Mellon University (CMU) [5, 6] implements the well known filter bank overlap addition method [7, 8]. However, other research have been conducted, introducing novel ways of increasing the environmental robustness to further decrease the WER [9–11].

Feature extraction is a fundamental part of ASR that has evolved over time. Different techniques have been used such as Perceptual Linear Prediction (PLP) [12], Mel-Frequency Cepstral Coefficients (MFCC) [13] and the tandem method [14] to name a few. These issues are however not further addressed as they do not directly relate to the main objective, stated in 1.

2.3 Hidden Markov Models

In ASR systems, Hidden Markov Models (HMMs) are used to solve the *decoding problem* $P_{max}(O, \Phi)$, i.e. finding the most probable hidden state sequence $S = (s_1, s_2, s_3, ..., s_t)$ in a HMM $\Phi(A, B)$ leading up to the sequence of observations $O = (o_1, o_2, o_3, ..., o_t)$, where A refers to the transitions probabilities and B the observable output probabilities of the HMM, as exemplified in Fig. 2.

A HMM refers to a graphical model utilizing some *hidden* stochastic Markov Process, meaning that the state sequence $S = (s_1, s_2, s_3, ..., s_t)$ leading up to the sequence of observations $O = (o_1, o_2, o_3, ..., o_t)$ at time t is unknown [15]. Furthermore, a Markov Process satisfies the Markov Property, referred to as memorylessness as the value of each state s_n is independent of all states preceding the previous state s_{n-1}.

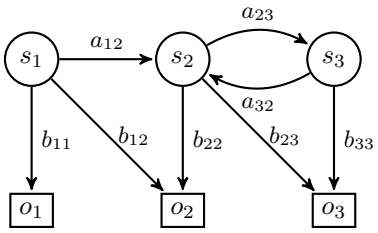

Figure 2: Example of a simple Hidden Markov Model; where s denotes a state, o a possible observation, a a transition probability and b an output probability.

2.4 Hypothesis Search

The problem of speech recognition can be formulated as

$$w^* = \underset{w}{argmax}[P(w|X)] \tag{1}$$

i.e. finding the most probable sentence w^* given an acoustic feature vector X. However, finding the probability $P(w|X)$ of a word w given a feature vector X

is a difficult problem. AMs are therefore modelled to solve the slightly easier problem of finding $p(X|w)$, the likelyhood of X given w. Together with the probability of the LM $P(w)$ and a pronunciation dictionary $\mathcal{D}$, mapping words to their phonetic transcription, the hypothesis search, or decoding problem, can be approximated as

$$w^* \approx \underset{w \in \mathcal{D}}{argmax}[p(X|w) \times P(w)] \tag{2}$$

the most likely combination of pronunciation and language rules.

2.5 Acoustic Model

The AM is responsible for transforming the spoken waveform into feature vector sequences. In Sphinx 4, the framework used for this project, there are three different types of AMs as part of the library: continuous, semi-continuous and Phonetically tied model (PTM). The PTM is the one that was used for this project and is also the default AM in Sphinx since it provides a good compromise between speed and accuracy, according to CMU, the developers of Sphinx [16].

The AM will not be covered in detail, but in short the AM is tasked to transform the acoustic pressure generated by, for example, speech into phonemes [17]. Speech consist of phonemes or other linguistic units, but understanding the acoustics of the variance created by such things as turbulent noise, voice breaks and tremors is key for an accurate transcription of the spoken word and is typically modeled as HMM triphones as illustrated in fig. 3 and exemplified in fig. 4. This This is what the acoustic model provides [18].

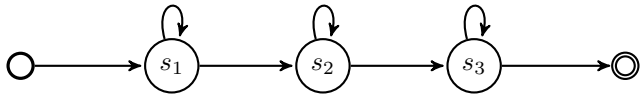

Figure 3: HMM Triphone

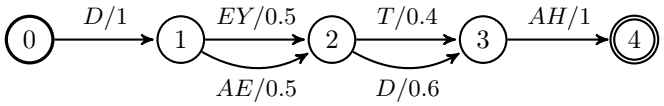

Figure 4: Finite automaton as a Pronunciation Model Acceptor built to terminate on different pronunciations for the word *data*.

2.6 Language Model

In ASR systems, the LM attempt to model the most probable ways of expressing one self in a particular language. When hypothesis search is conducted, the system calculates the likelyhood of different sentences and tries to predict the next word based on what was previously said. More formally, the LM models the probability $P(w_k|w_1, w_2, w_3, ...w_{k-1})$ of the next word w_k based on a the previously seen word sequence $w_1, w_2, w_3, ...w_{k-1}$.

2.6.1 n-gram Transducer Models

An n-gram is a sequence of n words, where a unigram denotes an n-gram with $n = 1$, a bigram with $n = 2$, a trigram with $n = 3$, a 4-gram with $n = 4$, a 5-gram with $n = 5$ and so on. An n-gram model is a Stochastic Language Model (SLM) for predicting an element in a Markov Chain of order $n - 1$ [19]. For example, an n-gram based LM could estimate the probability of a sentence $w = (w_1, w_2, w_3, ..., w_K)$ by applying the *chain rule of probability*

$$P(w) = \prod_{k=1}^{K} P(w_k|w_1, w_2, w_3, ..., w_{k-1}) \tag{3}$$

such that the probability of a single word occurring in the sentence approximates to the product of the probability of the $n - 1$ words preceding it [19]:

$$P(w) \approx P_n(w) = \prod_{k=1}^{K} P(w_k|w_{k-(n-1)}, ..., w_{k-1}) \tag{4}$$

Furthermore, an n-gram LM can be represented by a weighted Finite State Transducer (FST) such that the weights are the probability distribution of the n-grams [20]. FSTs are finite-state machines capable of translating, or *transducing*, the content of its input tape. If the input and content of the input tape match the conditions of a transition in the current state of the FST, a pop operation is performed on the input tape and a push operation on the output tape, effectively generating a transduced output string. FSTs can be used as the hidden model of a HMM where each accepted state sequence is a Markov Process which is what makes them suitable for language modelling in ASR systems. Eventually, each uttered word accounted for in the LM, AM and the dictionary will be evaluated like the transducer in Fig. 5.

In the simple bigram approximation FST illustrated in Fig. 6, the bigram $(w_1,\ w_2)$ is represented as a transition between state w_1 and w_2 where the weight of the transition between the two words is the probability of their bigram $P(w_2|w_1)$. The bigram (w_1, w_3) is however not represented in the model and is therefore estimated as $\beta(w_1) \times P(w_3)$ by the Back-off weight $\beta(w_1)$, the probability that w_1 is followed by *any* another word in the model, and the unigram weight $P(w_3)$, the probability that w_3 *exists* in a sentence [20,21]. This is done as to allow previously unseen sentence structures and combinations of

words that has been pruned to reduce the size of the model. Note however that the weights of n-gram FSTs are usually modelled as the negative logarithm of the n-gram probabilities $-log(P(w_k|w_1, ..., w_{k-n+1}))$ as to promote certain computations [20].

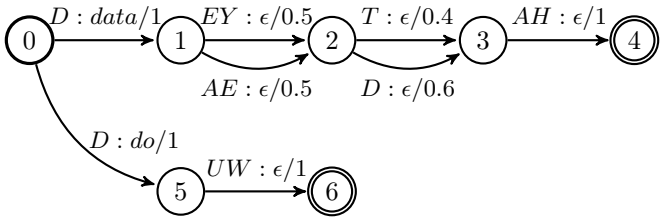

Figure 5: Pronunciation Lexicon Transducer as a Weighted FST built to terminate the words *data* and *do*

.

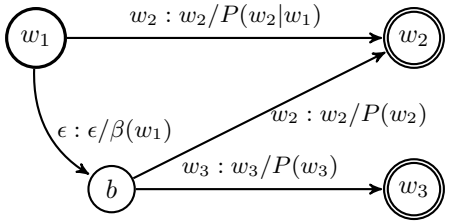

Figure 6: Bigram aproximation model [20]. As the bigram (w_1, w_2) exists in the model it has a probability of $P(w_1|w_2)$, whereas (w_1, w_3) has an estimated probability of $\beta(w_1) \times P(w_3)$ since it is not modelled.

2.7 Neural Networks

Language modelling is arguably a problem intuitively suitable for neural networks as their strength lay in finding an output based on complex, abstract relations in the input data. Due to this property, neural networks has been found to outperform purely SLMs, i.e. conventional n-gram models, in terms of accuracy [22, 23].

The design of the artificial neural network model is inspired by the structure of the biological brain and has had great success in solving complex problems as a tool for machine learning. As illustrated in Fig. 7 they are are constructed by an input layer x, an output layer y and a set of hidden layers h in between, each containing a set of nodes referred to as neurons. Typically, each neuron in such a net is connected to every other neuron of the previous layer by a weighted connection named synapse after its biological counterpart. Fig. 8 depicts the design of a neuron and how the weighted sum of its inputs $\sum_{i=1}^{n} x_i * w_i$ is passed

through an activation function f, typically a linear, sigmoid or logistic function, which determines whether or not the neuron should activate. In turn, activated neurons fires a signal through their weighted synapses to the neurons of the succeeding layer, and such is the signal propagated through the network.

Neural networks are commonly used for solving different classification problems e.g. persons based on human facial features, abnormalities in large data sets or the next element of a sequence, like *words in a sentence* for instance. Classification networks can be taught to distinguish different classes by conducting supervised learning. In supervised learning, a training set is used, stating what input corresponds to what output label. As the inputs are propagated through the network, the outputs are compared to the correct labels in the corpora and the parameters in the network are adjusted to improve accuracy of future predictions. The most common way to do this is a process called backpropagation. It does so by propagating the error backwards from the output nodes, through the hidden layers and back to the input nodes, adjusting each synapse parameter by optimizing an objective function. For example, this could be done by minimizing a cost/loss function such as the mean squared error

$$C(w, b) = \frac{1}{n} \sum_{i=0}^{n} |\hat{y}_i - y_i|^2 \tag{5}$$

where the cost function C with weights w and biases b is calculated as the mean of the error $\hat{y} - y$ squared, such that $\hat{y}$ depicts the vector of n predictions and y the observed outputs.

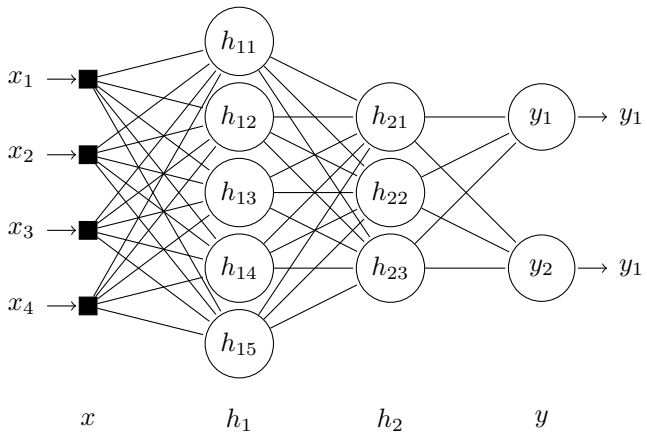

Figure 7: Neural network with input layer $x = \{x_1, x_2, x_3, x_4\}$, two hidden layers $h_1 = \{h_{11}, h_{12}, h_{13}, h_{14}, h_{15}\}$, $h_2 = \{h_{21}, h_{22}, h_{23}\}$ and an output layer $y = \{y_1, y_2\}$.

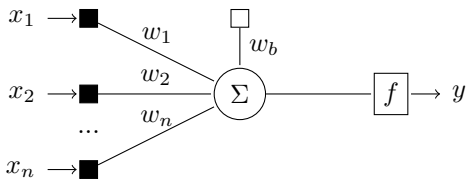

Figure 8: A neuron calculates the weighted sum as the dot product of its input $x = \{x_1, ..., x_n\}$ and synapse weights $w = \{w_1, ..., w_n\}$, including the bias weight w_b, and passes the summation as input to its activation function f which determines whether or not the neuron should fire.

2.7.1 Neural Networks Based Language Models

The Neural Language Model (NLM) can be considered a classification network where each word in the models vocabulary is its own class. The current word is thus represented by some input vector, e.g. a one-hot representation, and the output vector, i.e. the next word, is the probability distribution between the different classes. By conducting supervised training, a network can thus effectively learn what word, or class, is likely to follow another by minimizing C, the cost of the negative log likelyhood function where y are the label vectors and $\hat{y}$ the observed predictions of the network:

$$C = -\sum_{i=1}^{n} y_i \times log(\hat{y}_i) \tag{6}$$

However, traditional neural networks lacks *memory*, the ability to remember previous predictions is intuitively important when building neural language models. A one-hot word level language model without memory would simply propagate each word w_t of a sequence $w = (w_1, w_2, ..., w_n)$ through the network, unable to account for the previously propagated words, effectively calculating and forgetting the probability of each bigram $P(w_{t+1}|w_t)$ instead of calculating the probability of the entire sequence $P(w_n|w_1, w_2, .., w_{n-1})$. This issue can be solved by introducing feedback loops into the network, thus allowing information to persist over the execution of a number of predictions. Such a construction is called a Recurrent Neural Net (RNN).

2.7.2 Recurrent Neural Networks

Fig. 9 is an alteration of Fig. 7 illustrating how nodes in the recurrent hidden layer has a box attached, they represent a bounded history, a loop of previous states being fed back into each node. To illustrate this feedback loop one could *unfold* the history where s_t is the state at time t and its calculation,

$$s_t = f(wx_t + ws_{t-1}) \tag{7}$$

is based on the previous state of the node, as well as the input at the current step.

As seen in Fig. 10 each element in the history buffer is fed as another input to the neuron, thus allowing it to fire based on a sequential context.

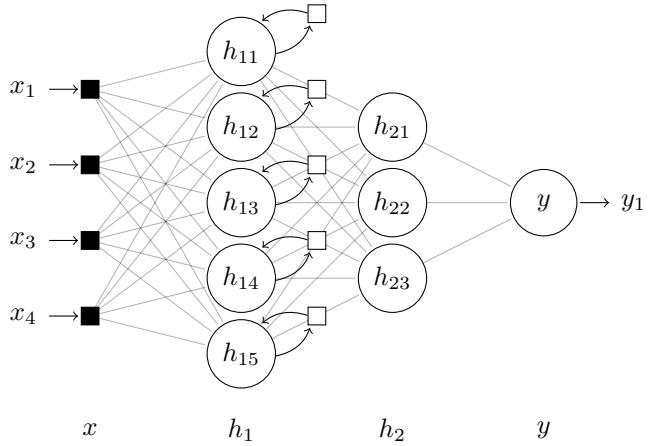

Figure 9: Recurrent Neural network with input layer $x = \{x_1, x_2, x_3, x_4\}$, a recurrent hidden layer $h_1 = \{h_{11}, h_{12}, h_{13}, h_{14}, h_{15}\}$, a non-recurrent hidden layer $h_2 = \{h_{21}, h_{22}, h_{23}\}$ and a single output in its output layer y. Each box attached to the nodes in h_1 represent the nodes recurrent memory.

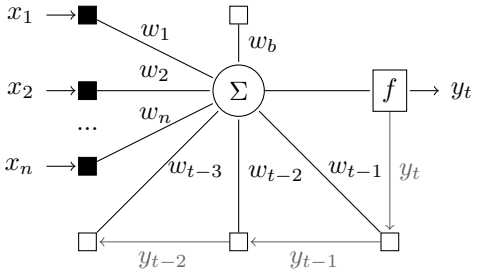

Figure 10: A neuron in a recurrent neural network with a memory of size 3 uses the history $(y_{t-1}, y_{t-2}, y_{t-3})$ to calculate y_t at time t.

2.7.3 Long Short-term Memory

Even though RNNs solves the problem of data persistence and could be trained to find the next word *Stockholm* given the history *"the capital of Sweden is"*, modelling RNNs to remember long term dependencies has been proven difficult [24]. For example, the city of Stockholm should arguably be more likely to be uttered in a conversation about Sweden than other counties, *even when they occur sentences apart*. A network design based on RNN utilizing Long Short-Term Memory (LSTM) algorithms is however able to learn long term dependencies between input data and base predictions on the *topic* of conversation as well sentence structure [25, 26].

The older the history is of an LSTMs, the lesser is its contribution to the output of the model. The gradually diminishing effect of the information applied from the context history can be exploited in neural LMs using recurrent layers by restricting the memory size in an n-gram fashion [27]. This does not only mean that the number of hyper-parameters in the RNN can be decreased but also that there is no point in conduction hypothesis search on sequence lengths longer than the context history. This means that all predictions that share the same context history consisting of at most $n-1$ words are truncated. Fig 11 illustrates how two full search-tree histories shares the most resent history "sell at" and is therefore considered equivalent when applying trigram history clustering. Due to the truncation of n-gram based history clustering, the same search heuristics commonly used for n-gram FST models can be applied for neural LMs [27] e.g. beam-search.

The issue of neural LMs and their computational costs has been addressed in several studies with the aim to utilize DNNs for direct lattice rescoring. As a lot of computational cost stems from output layer computations, one approach is to model the output layer with fewer words than present in the input vocabulary, a so called short-list [28]. This approach requires an output node representing out-of-shortlist words i.e all words *not* represented by a node in the shortlist, without it, out-of-shortlist word statistics are simply discarded.

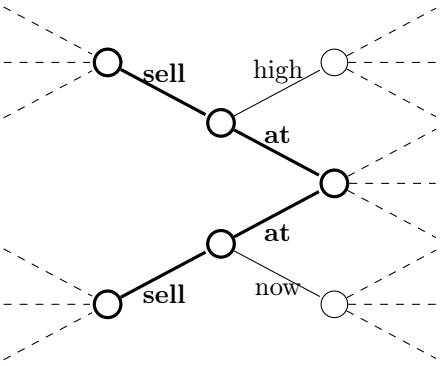

Figure 11: An example of trigram based RNNLM history clustering [27].

2.8 Language Model Interpolation

Neural networks have proven successful when used in conjunction with more traditional HMMs. The downside of using DNNs, however, is that the training time of a model of a respectable size on a corpus of a respectable size becomes substantial. The application area in which new words occur on a regular basis makes it impractical to allocate the resources needed to retrain the models as often as new words appear. A solution to this problem is to interpolate two different language models, trained on different corpora, one larger generic n-gram model and one smaller domain specific neural net based model. The upside with this approach is that both training and execution time of the NN model can be reduced significantly.

Interpolation of two models in ASR systems are often done linearly, as such, interpolation of a generic n-gram model and a domain specific neural network model could be described as

$$P(w_0^k) = \alpha P_{NG}(w) + \beta P_{NN}(w) \tag{8}$$

where k is the length of the word sequence and $\alpha + \beta = 1$ are weights that is manually tuned to produce the best result.

However, the runtime of large neural network models makes it an unviable option to utilize linear interpolation for direct lattice rescoring without taking additional measures. Thus is the domain specific neural network model often used to select the best hypothesis out of an *n-best list* from the generic model [23]. This approach is however highly use case dependent and not fully reliable unless all keywords in the domain specific model overlaps with the vocabulary of the generic model, an issue addressed in section 4.2.2. Furthermore, there is no guarantee that the best or even a good prediction of the domain specific model is among an n-best list conceived by the generic model.

3 Related Work

ASR is fundamentally addressing a pattern recognition challenge that can be very difficult to automate [29] [30]. These types of problems appear to necessitate intelligence to solve, especially since they seem trivial for humans while immensely difficult for machines. As such involving techniques that emulate the human brain like artificial neural networks have shown great potential in solving them [30] and will be attempted in this project.

When talking about NER it's common to define a set of characteristics, called features, that a proper noun should have in order to be considered a word of interest [31]. The term *proper noun* stems from the fact that given the right context anything can become a noun. For example the word *blue* is a color but could also be used as the name of a company. Another example is that without the right context a proper noun would be considered nonsense. For example *Netflix* is not a word in any dictionary, but should be found and considered a noun in the context of market surveillance. NER has proven to

be a useful tool to solve these kinds of problems [32]. While these aspects are interesting and should be considered a good candidate for future work, it was decided to not pursue this approach when tagging instruments in the corpus material, as discussed in section 4.2. Since all instruments in the corpus was known beforehand, it was not of interest to implement these kinds of feature based NER system at this time and choose the more simplistic approach of tagging the corpus our selfs and implement our own NER algorithm.

4 Method

By utilizing an open-source ASR system, Sphinx4, the intention is to build a proof-of-concept AVTS system that produces data which can be combined with structured Market Data for the purpose of market surveillance. By configuring the AVTS system to include the phonetic transcription of the instruments that together constituted the OMX30 index found in Appendix F, based on the pronunciation dictionary found in Appendix E, various models will be evaluated on their ability to transcribe four recordings in the financial domain.

The relationship of interpolation between domain-specific LMs and a generic LM will be explored by evaluating the WER as expressed in Equation 9 and NER in terms of F_1-score as expressed in Equation 10. WER provides a measurement of how well a model is able to transcribe recorded speech and is defined as the substitutions S, deletions D and insertions I divided by the number of words in the reference transcription N.

$$WER = \frac{S + D + I}{N} \tag{9}$$

The F_1-score measures the accuracy of binary classification problems like NER and provides a measurement of how well the models are able to detect instruments for effective market surveillance. It is the harmonic mean of the models transcription *precision* and *recall* i.e. the number of true positives TP divided by the number of recognised Named Entities (NEs), including false positives FP, respectively the number of true positives TP divided by the number of all NEs that should have been identified by the system, including false negatives FN. A perfect F_1-score is therefore 1, whereas the worst possible score is 0.

$$F_1 = \frac{2TP}{2TP + FP + FN} \tag{10}$$

The WER of the generic model will provide a baseline for transcription accuracy but is expected to be fairly low on the domain-specific recordings. Likewise, the small size of the domain-specific models should prove to be somewhat ineffective at translating the same recordings but with the ability of NER. The thesis to be explored is whether interpolation of these models provide a better result in terms of WER and the ability to detect NE and draw conclusions about the relationship between interpolations weights and these measurements.

The interpolation thesis will be considered successful if the WER of the interpolated models are lower than the generic model and the domain-specific models in isolation, provided that at least 75% of the NE are found in the test recordings. The AVTS system will be considered successful in itself as a proof-of-concept if a WER of 25 or below and an F_1-score of 0.8 or higher is achieved.

4.1 Delimitations

A vital part an AVTS system utilizing ASR is to transcribe dialogues between people within a certain setting, for instance a trader and a stock broker talking about a trade or trading some particular instrument. Since there is no reliable source of transcribed recordings available for the authors, a problem arises. In order to create a fair, non arbitrary, test for the system such material must be acquired. If there were a body of such material, whether of traders trading or otherwise, then the material must be transcribed by hand. This is of course a very large undertaking within itself and deemed, by the authors, not within the scope of the project. These problems and how they are solved or circumvented are discussed further in section 4.2.

The project will mainly focus on recognizing instruments not part of any corpus. However, the world is global and within the financial market it is not only possible, but probable even, that instruments may have foreign names that has no phonetic transcription available, which in addition is highly likely to vary depending between different speakers. This within itself is it's own research area, well worthy of it's very own Master's Thesis project, and is thereby assumed to be correctly working.

Secondly the project is relying on the quality of the corpora used for the two different language models. The material collected for this project is part of a proof of concept and will not be part of any release. This means that the WER should not be the only thing that determines the success of the project, but this will be discussed more in detail in sections 5 and 6.

4.2 Corpus Composition

In order to develop, tune and test the implementation, a domain specific corpus had to be generated. As discussed briefly in section 4.1, there are some challenges associated with this. Firstly, the success of the tests are highly dependant on the relation between the corpus material and the voice recordings used to test the program. Since these two entities are separate, and not part of the same corpus, the outcome can be biased. For example, if the voice recordings contain a certain sentence, which differs widely from anything in the corpus material, the outcome will be very poor. However, if that particular sentence is represented exactly within the corpus then the tests will yield very good results. This is a problem and since the goal for the overall project, exceeding that of the Master's Thesis, is to create a domain specific language model that can handle the type of domain specific banter and such. In consultation with a domain

expert, and with some research, a couple of sentences to test the application was produced and recorded, and a source for domain specific material was established. This source consists of a collection of web forums that was scraped and processed for language model creation. As the vigilant reader will notice, this does not withdraw anything from the risk of biasing the test result. Given the poor circumstances, in this regard, there is no satisfying way to eliminate this risk, however, in an attempt to reduce it some measures was taken, further discussed in section 6.

Gathering a corpus was done by using a web crawler to scrape different forums and sites, with domain specific content. The specific sites were used for data collection was selected in joint consultation with Scila AB. The data needed to be domain specific but have some overlap with the generic language model corpus. It needed to contain words that are used in a domain specific way, perhaps by using a verb as a noun, for example "Blue point" where "blue" is used as a noun and not a verb, but also words not contained at all by the generic model. This could be guaranteed to be the case since all words in a language model must be represented in the dictionary, which means that a word not found in the dictionary is not part of the vocabulary of the language model.

4.2.1 Test Data

After scraping the websites selected for corpus construction, a selection of articles were excluded from the corpus material, and set aside for another use. Instead of using these articles as part of the training material, they were used as base for voice recordings to be used to test the entire system, discussed more in section 5. The files were not used exactly as is to test the system, but some of it's key features were altered to test different aspects of the system. There are two main aspects of the system that is critical to the projects success that was tested; numbers and instrument names. The test files are by no means exhaustive, created to test a small portion of the system. In order to get a varied test the numbers were tailored to give a broad spectrum of what might be heard in a live situation. For example, fractions, decimal numbers and whole numbers are all represented, but not all variations are. The same principal was applicated to instrument names with a couple of restrictions, further discussed in section 5.

The articles selected for testing was recorded by four native US English speaking individuals. Two female and two male students were recorded, with little or no accent.

4.2.2 Information Abstraction

LMs used in ASR systems can be subjected to information abstraction for a number of reasons. The AVTS LMs are modelled with a set of different abstractions as to improve their accuracy and allowing infinite additions of certain types of entities. Abstraction is done by classifying words according to some desirable criteria and the LM corpus is then processed such that the classified

words are replaced by a word, or *tag*, representing the class.

Out-of-Vocabulary Words Words that only occur a few times in the training corpus of a LM makes the model less effective. They do so by enlarging the model without providing it with any reliable way to predict the seldomly uttered words as there is not enough statistic information in the corpus about them. In extension, this has an effect on the entire ASR system. More words means larger HMM search trees and longer probability calculation times. However, one can classify seldom used words as *unknown* and replace them in the corpus with a common tag $< u >$. The effect of this abstraction is two fold, the size of the model is reduced in relation to some used occurrence threshold and the model gains the notion that *there exists a set of words outside of its vocabulary*. Even though such Out-of-Vocabulary (OOV) words may not contain a lot of information in themselves compared to words in the vocabulary, such an input node allows the model to handle an infinite number of input words and, depending on implementation, could make contextual predictions without having learned every single word of a sequence.

When interpolating LMs in ASR systems there are two choices in terms of the vocabulary used to conduct hypothesis search. Either the search is conducted based on the intersection or union of the LMs vocabularies and unless all models uses the same vocabulary, the united vocabulary will provide the ASR system with a more complete set of available language structures. However, a united vocabulary can only be used as long as there is some mechanic in the ASR system that keeps the models from trying to predict the probability of OOV, as they are unable to do so by design. If a LM receives a word sequence containing a word from another LMs vocabulary during hypothesis search, the OOV word can be exchanged with the $< u >$ tag, allowing the LM to make a prediction.

Named Entities The concept of OOV words is expanded and further developed in the AVTS system to hold all words of a certain type under one tag. By doing this the ASR recognizer can be set to recognize new words within a specific category set by the tag. This generalises the probability over all members of that category. For example, if numbers are represented by a tag and the corpus contains the following sentence *"Give me five apples, please"*, then that sentence would be rewritten as *"Give me* $< n >$ *apples, please"*. This way, any number can be inserted instead of the $< n >$ tag, all with equal probability. This is beneficial when the actual number in the text is unimportant but the fact that there is a number in a particular position in the word sequence is of importance. For example, if the speech to be transcribed is suspected to cover a large amount, for some definition of large, of talk about apples and numbers of apples and these numbers are of particular interest, then it would make sense to insert a tag for the number in the text corpus to remove bias from certain numbers.

All of this is then later accommodated for when creating the HMM search tree. The model reader must be created in such a way that when loading the statistical language model no tags are loaded into the vocabulary. The vocabulary is a data structure holding all available words for a particular LM and made available for other parts of the program in order to have a search

conducted. At a later stage when the search engine asks for the probability of any word that is a member of a tag, then the probability of that tag is returned.

Numbers A statistic language model is a representation of the statistical connections of words and sentences. However, not all words are valuable in it's explicit form but more so in it's abstract form. For example, in the sentence *"I'll sell you eighty shares"*, the number eighty is not valuable. The sentence represents someone offering someone else equity in a company, that number happens to be eighty, but could just as well be anything else. It is not uncommon that people trade a number of items with one another, but that number varies, and should not be skewed towards any specific number arbitrarily set in the corpus material. To alleviate this problem a number tag $< n >$ is introduced, and all numbers in the corpus is replaced with this numbers tag, resulting in the following sentence, *"I'll sell you* $< n >$ *shares"*.

Furthermore, creating a dictionary and vocabulary with all numbers in the world is impossible given that there exists an infinite amount of numbers. However, all numbers consists of a finite set of words when spoken. All the spoken numbers in e.g. 123, "one hundred twenty three" or "one hundred and twenty three", can be substituted with an abstraction and you will get $< n >< n >< n >< n >$ or $< n >< n >$ *and* $< n >< n >$. If numbers, however large, are parsed to a single $< n >$ tag the corpus will skews the word search results in relation to the unprocessed corpora however the authors claim that having a single $< n >$ tag for all numbers not only makes processing of the corpora a lot less complicated but also does not skew the resulting model into providing an extreme bias towards numbers following numbers.

Instruments & Companies Following the reasoning about numbers uttered in conversation, companies and financial instruments can be classified and abstracted as a single tag $< i >$ to represent that any company or instrument can be spoken about in the same way. This is of course a trait of the domain of market surveillance that does not necessarily hold for systems transcribing speech of other domains. Using the tag representation also has the advantage that the system allows addition of companies and instruments continuously as they enter the market by appending them to the pronunciation dictionary instead of gathering corpora containing the new information and retraining the model.

However, names do not necessarily adhere to the standard spelling conventions of a particular language due to them being of foreign origin or other factors such as the creative mind of a parent or a company founder. This inherent ambiguity of pronunciation becomes problematic when designing a system building name hypothesis of companies and financial instruments. An example of such an ambiguity is the Swedish *Kinnevik B* stock. It is unclear exactly how a non-Swedish speaker would pronounce Kinnevik and whether a native Swedish speaker should use a Swedish pronunciation, CH IH N EH V IY K, in a conversation with a non-Swedish speaker such as an Englishman to be understood or an English variant e.g K IH N V IH K. Constructing pronunciation dictionaries is a non trivial task as shown by this example and an automated approach using Grapheme to Phoneme (G2P) converters thus becomes a research area on its

own. Therefore, the AVTS system uses a hand made pronunciation dictionary for the instruments in the OMXS30 index, an index of the thirty most traded stocks on the Swedish market venue Stockhlmsbörsen.

Consequences One of the problems with creating statistic language modeling is the volatility of the corpus material used. Introducing an abstraction for named entities reduces that volatility, and takes the model closer to a grammar like state. Using a language model that consists purely of grammar is not desirable since no official language follows any strict grammar, which makes them impractical to implement, and because people do not always follow the grammar, rendering the LM useless. So how many new tags can be introduced? For example, the sentence "Give me five apples, please" could be made abstract by using tags, resulting in "$< verb > < noun > < n > < object >, < adverb >$". This way the corpus is formed more as a statistical grammar than anything else. There are mainly two problems with this. Firstly, no language, English included, follows any strict grammar. For example, "Drive house three cups, truthfully" is not a meaningful sentence but does not violate the rules set above. Secondly, it quickly becomes impractical to replace all words with it's correct word class or any other grouping. Not only because of the mentioned problem, but also because all words classified needs to be divided and correctly inserted in the appropriate list for later search. As a further consequence of this the searches would be much more time consuming. The searches gets larger because every time a tag is encountered, all the values that has been abstracted needs to be looked up.

Since an AVTS system lies in the domain of financial markets, the decision was made to limit the number of tags to instruments, a few names and numbers. The reason why these entities are abstracted away from the corpus material is because they play an integral part of market surveillance but are without value in their explicit form as a corpora.

4.3 Implementation

The AVTS system is developed as an extension to the existing open-source ASR library Sphinx 4, developed by CMU. The three main parts that was created was the added support for language models using neural nets, a parser for cleaning up corpus material and the added functionality of using tags to identify NEs, conceptually discussed in section 4.2.2.

As seen in the system architecture diagram Fig. 12, the AVTS system utilizes a Web Crawler that scrapes the internet for raw corpus material c and extract the relevant data $\hat{c}$. $\hat{c}$ is then passed to a Parser which refines the corpora before it is stored for later use. This final version of the corpora, c^*, is used to generate statistical and neural LMs which are interpolated in the search for the word sequence w^* based on the utterance u stored in the voice trading database.

The hypothesis search tree in ASR systems utilizing LMs with a respectable vocabulary size tends to become too large for exhaustive search approaches, such as the breadths first Vertibri algorithm, to be deemed as feasible options. Optimized methods such as beam search can thus be used to find a solution

within a reasonable time frame and was thus selected for the AVTS model, however sacrificing the guarantee of finding the optimal solution.

The interpolation of different models is done by adding the weighted probability of each given word sequence and configured model. This means that the different models can be weighted to calibrate the result. If a word in the current hypothesis search sequence is unknown to a specific LM in the interpolated LM it is exchanged with the the unknown token $< u >$ for that model.

4.3.1 Web Crawler

As a mean to attain corpus material, the Java based web crawler library crawler4j was utilized. Each web crawler is designed to target a specific source and consists of two classes, the *web crawler* and its *controller*. The controller class specifies on which website search should be conducted and how many crawler instances should work in parallel to fetch data from it and where to store the gathered information. Each crawler class is specilaized to gather data on a specific website by implementing two methods, *shouldVisit()* which decides which hyper links to follow continue the search and *visit()* which processes the web page's html elements to extract the relevant information.

4.3.2 Corpus Parser

To make the extracted material of the web crawler suitable as language modeling corpus it must be further processed. The most important aspect of the corpus parsing is to normalize different variants of words, implement the various information abstractions discussed and structure the data in a suitable format. Following are the operations executed by the corpus parser:

Add start and stop tag to each sentence w in the corpus C.

$$\forall\{w \in C\} : w \leftarrow [<\text{s}>, w, <\text{/s}>] \tag{11}$$

Convert currency symbols to words.

$$\begin{gathered} \text{Given the currency conversion map} \\ f : A \rightarrow (\text{dollar}, \text{pound}, \text{euro}, \text{yen}) | A = (\$, £, €, ¥) \\ \text{then} \\ \forall\{s \in w | s \in A \wedge w \in C\} : s \leftarrow f(s) \end{gathered} \tag{12}$$

Abstract instrument names and symbols to related NE tag.

$$\begin{gathered} \text{Given the set of instruments } I \text{ then} \\ \forall\{i \in I | i \in w \wedge w \in C\} : i \leftarrow <\text{i}> \end{gathered} \tag{13}$$

Abstract person names to related person name tag.

$$\begin{gathered} \text{Given the set of names } P \text{ then} \\ \forall\{p \in P | p \in w \wedge w \in C\} : p \leftarrow <\text{p}> \end{gathered} \tag{14}$$

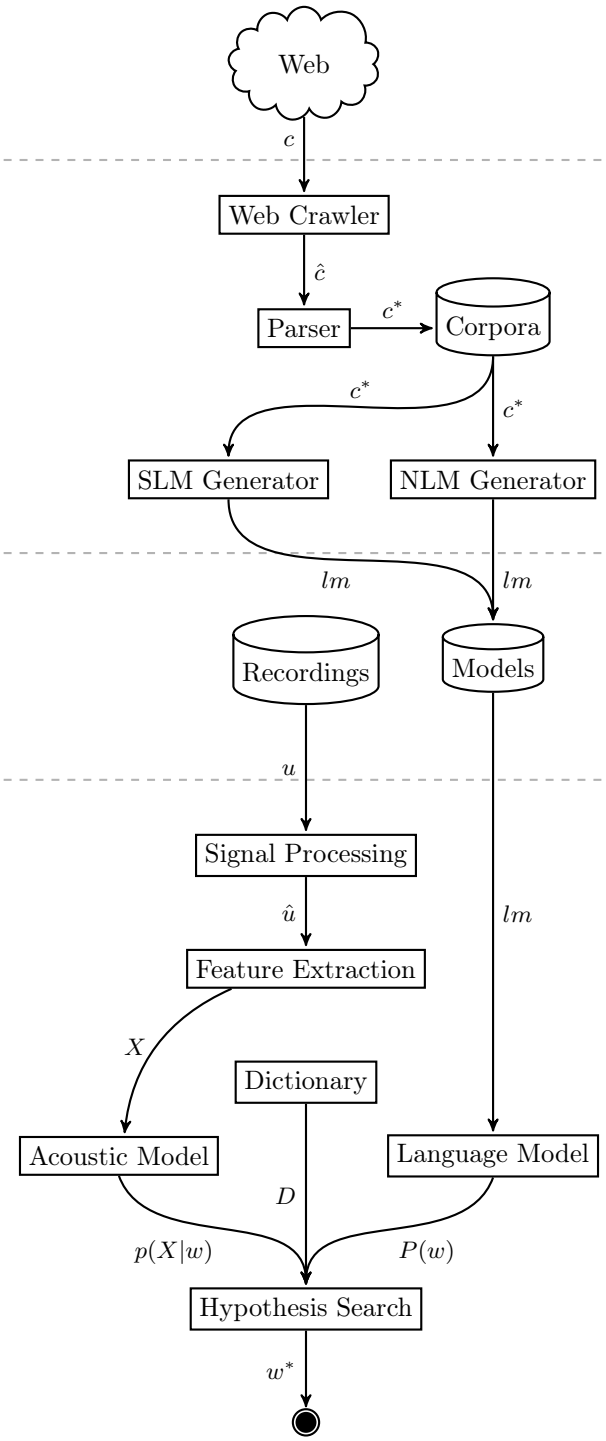

Figure 12: AVTS System Architecture

Abstract numbers.

$$\begin{gathered}\text{Given the number conversion map} \\ \forall\{n \in \mathbb{R}\} : n \leftarrow \text{<n>}\end{gathered} \tag{15}$$

Abstract infrequently occurring words as OOV.

$$\forall\{w \in C_n \mid \sum_{i=1}^{n} [w_i = w] \leq 5\} : w \leftarrow \text{<u>} \tag{16}$$

Table 1: Data set

Data set	sentences	word count	<u>	<i>	<n>	<p>
Training	43148	867206	19898	12701	30790	4997
Validation	4795	95711	2094	1220	3315	619

4.3.3 Stochastic Language Model

LMs could now be constructed given the output described in the previous section 4.3.2. For this project the SLM variant n-gram LM was chosen, not only for it's popularity but also for it's simplistic nature.

The n-gram LM consists of the frequency by which word sequences of length 1 to n is present in the provided body of text. This means that when n increases the statistical probability for any given word sequence lowers, indicating that the relationship between the size of the corpus material and the n can be optimized. This relationship does not fall under the scope of this project.

In order to create the SLMs intended for this project the tools supplied by CMU Sphinx4 in the ARPA format was utilized. These tools provided the means for creating multiple LMs with different n-values and corresponding dictionaries, later used by the system.

4.3.4 Identifying Named Entities

In order to make the system as modular as possible, the amount of tags used is specified only during execution. This means that the number of tags used is specified while creating and managing the corpus, used as the domain specific one. The system builds up a search tree using a vocabulary. The vocabulary consists of all the unique words in the model. These words are then fed back into the HMM and their probabilities fetched from the language models. The problem is that since all the tags supported are scattered within the model, these needs to be replaced in the vocabulary, one cannot get the probability of a tag within speech. So all tags are removed from the vocabulary, and instead all the members of all the tags are added. All members are marked with the specific tag that it is a member of. At a later stage when a request for a probability is received, these members are switched back to the tag within the word sequence.

This means that all the members get the same probability from the language model. Using the previous example in section 4.2.2, the number "five" in the sentence "Give me five apples please", is just as likely as the number "four". In this example it is easy to argue that a smaller number is more likely than some large arbitrary number like one hundred, or a billion. In regards to the number, there is a case to be made for this, however, it is not as clear in regards to other tags, instrument names for example.

4.3.5 Neural Net Implementation

The Sphinx 4 library does not support *any* neural network features and as such, does not have native support for NLMs. However, this feature was implemented in the ASR system by modelling neural networks with the Java library Deeplearning4j (D4J) and implementing custom Sphinx 4 LM classes that utilizes said models in the speech recognition process. An RNN model illustrated in Fig. 13 was created using the parameters seen in Table 2 which yielded a model that yielded a 37.8% improvement on the training set, plotted in Fig. 14, and 8.7% on the validation set, plotted in Fig. 15, over 60 epochs. The neural network was trained using negative log likelihood as loss function which aligns with how the SLM expresses n-gram likelihoods. Alignment between how the models express likelihood is crucial for appropriate interpolation.

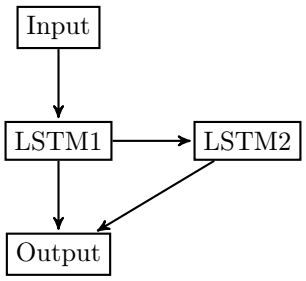

Figure 13: LSTM Model Setup

Table 2: LSTM Model Parameters

Parameter	Value
Layer size (LSTM 1)	2500
Layer size (LSTM 2)	2500
Optimization Algorithm	Stochastic Gradient Descent
Truncated back propagation through time length	35
Example length	70
Learning rate	1.0E-5
RMS Decay (Root Mean Square Decay)	0.95
L2 Regularization (Weight Decay)	0.001
Loss Function	Negative Log Likelihood

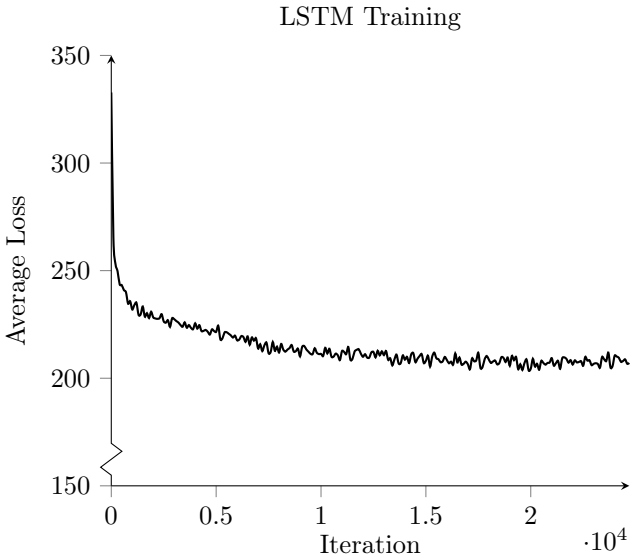

Figure 14: The NLM model gains the most knowledge in around the first 5000 iterations, about 12 epochs, of learning.

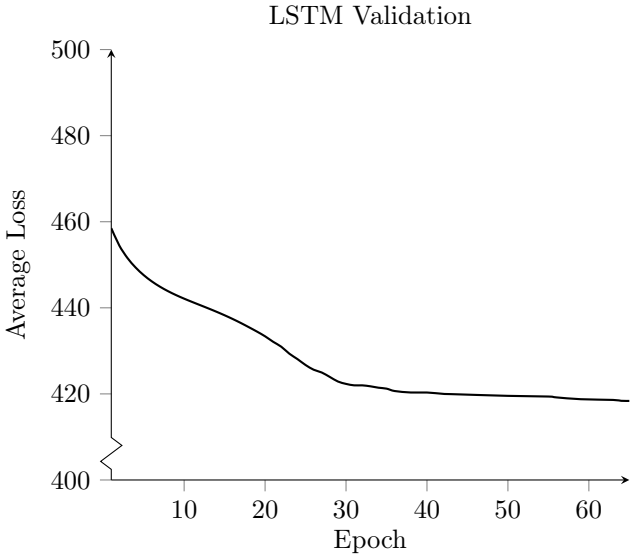

Figure 15: The NLM model is slowly improving on the validation set over 60 epochs. Improvement is leveling out around 40 epochs.

5 Results

The AVTS system was tested using the files discussed in section 4.2.1. The accuracy in terms of WER and NER can be found in the tables Table 9 - Table 23 highlighting the results for the various model/weight combinations.

Running the generic model on the four audio recordings provided an average WER of 31.63 as shown by Table 3. The model was run both as a standalone and as a single model interpolated with no other model to show that the interpolating code does not influence the results. Table 4 also shows that, as expected, none of the named entities were found. This is intended to show that the domain specific models are the only models contributing to find named entities without any additional processing of the transcriptions.

Table 3: Word Error Rate *generic*

Model	Time	Rec 0	Rec 1	Rec 2	Rec 3	Average Score
generic	1.14	30.50	23.42	31.22	41.38	31.63
generic − interpolated	1.03	30.50	23.42	31.22	41.38	31.63

Table 4: Named Entity Recognition *generic*

Model	Weights	Rec 0	Rec 1	Rec 2	Rec 3	Sum
generic						
true positives		0	0	0	0	0
false positives		0	0	0	0	0
false negatives		2	2	8	8	20
generic interpolated	50/50					
true positives		0	0	0	0	0
false positives		0	0	0	0	0
false negatives		2	2	8	8	20

5.1 Stochastic Model Results

The results of running the statistical-based *tagged n-gram* models on the test recordings can be found in Table 5, a summary of Table 9 and Table 10 found in Appendix A. The models are named on the format $tngX_Y$ where X refers to the models n-gram size and Y the depth used in the hypothesis search algorithm.

It can be observed that the *tng* models correctly transcribes about two out of three words with a WER close to 30, slightly lower than the generic model benchmark. As shown in Table 10 the models found 19 of the total 20 named entities. However, the models are biased towards finding named entities as illustrated by the 12-14 false positives in the transcriptions which in turn has an effect on the $F1 - score$, where the best models had a score of 0.7547. It should be noted that the smaller trie-gram models performed the best in both WER and $F1 - score$.

Table 5: Results *tng*

Model	WER	F_1
$tng3_3$	**29.97**	**0.7547**
$tng4_3$	30.26	**0.7547**
$tng4_4$	30.39	0.7273
$tng5_3$	**29.97**	**0.7547**
$tng5_4$	30.39	0.7273
$tng5_5$	30.61	0.7273
$tng6_3$	**29.97**	**0.7547**
$tng6_4$	30.68	0.7273
$tng6_5$	30.61	0.7273
$tng6_6$	30.67	0.7273

5.2 Neural Net Model Results

Table 6 contains a summary of the result tables Table 11 and Table 12 found in Appendix B. Compared with the results in Table 5 the NLM model performed significantly worse in terms of WER and NER relative to the SLM counterpart using the same search depth. This can be illustrated by comparing the best model, $tng3_3$, with a WER of 29.97 and $F1 - score$ of 0.7547 against the best performing NLM configuration, nlm_5, with a WER of 40.90 and $F1 - score$ of 0.7143, which is worse than the worst performing tng models as well as the generic model benchmark of a 31.63 WER.

Furthermore, provided that CPU:s were used to perform neural network activation's calculations, instead of GPU:s, the recordings were transcribed by a magnitude slower than the hash set based implementation of the SLM. While this comes to no surprise to the authors, due to the nature of neural net calculations, it is worth noting the significant increase in runtime as the hypothesis search depth increases.

Table 6: Results nlm

Model	WER	F_1
nlm_3	41.00	0.6897
nlm_4	40.95	0.7018
nlm_5	**40.90**	**0.7143**
nlm_6	41.15	**0.7143**

5.3 Interpolated Model Results

Table 7 shows the domain-specific models examined with the same search depth as the size of the generic trie-gram model. The $tng3_3$ model gets a better WER and F_1-score when interpolated with the generic model. While the nlm_3 model also gets a clear improvement compared to running the model by itself it does not perform any better than running the SLMs.

A slightly better WER than running the $tng3_3$ model alone was achieved by interpolating the $tng3_3$ with the nlm_3 configuration as seen in Table 7, however with a runtime penalty. This inspired the three model interpolation $tng3_3/nlm3/generic$ which can be found in Table 8. Combining the two different types of domain-specific models with the generic model in a 25/25/50 percent split yielded the best result in terms of WER, with a 25.08 WER, wheres the best model in terms of NER was the 5/5/90 percent split with an F_1-score of 0.9091.

The trie-gram model, five-gram and six-gram models had similar, yet difficult to compare, WER when used to transcribe the recordings on their own. The reason for this is because the models performed slightly better then the other on different recordings. However, a more noticeable difference appear when interpolated with the generic model. As seen in Table 13 and Table 14 in Appendix C, the trie-gram models clearly outperforms the larger domain-specific n-gram models when interpolated with the generic model. The larger

domain-specific n-gram model is used alone for scoring word sequence searches longer than three words due to the generic model being a trie-gram model. The conclusion being that that you do not gain a better WER using a relatively small, and therefore biased, domain-specific model with a deeper depth than your larger generic model when interpolating the two.

Table 7: Interpolated Model Results

Model	Weights	WER	F_1
$tng3_3/generic$	10/90	27,77	**0.8000**
$tng3_3/generic$	20/80	26,81	**0.8000**
$tng3_3/generic$	30/70	26.52	**0.8000**
$tng3_3/generic$	40/60	26,07	**0.8000**
$tng3_3/generic$	50/50	**25,75**	**0.8000**
$tng3_3/generic$	60/40	25,87	**0.8000**
$tng3_3/generic$	70/30	26,08	**0.8000**
$tng3_3/generic$	80/20	25,97	**0.8000**
$tng3_3/generic$	90/10	26,44	0.7843
$nlm_3/generic$	10/90	30.93	0.7407
$nlm_3/generic$	20/80	30.41	0.7273
$nlm_3/generic$	30/70	29.84	0.7407
$nlm_3/generic$	40/60	29.62	0.7407
$nlm_3/generic$	50/50	29.74	0.7547
$nlm_3/generic$	60/40	29.93	0.7692
$nlm_3/generic$	70/30	30.60	0.7547
$nlm_3/generic$	80/20	32.04	0.7547
$nlm_3/generic$	90/10	33.42	0.7547
$tng3_3/nlm_3$	10/90	35.04	0.7407
$tng3_3/nlm_3$	20/80	33.69	0.7843
$tng3_3/nlm_3$	30/70	32.64	0.7843
$tng3_3/nlm_3$	40/60	31.55	0.7843
$tng3_3/nlm_3$	50/50	31.13	0.7692
$tng3_3/nlm_3$	60/40	30.67	0.7692
$tng3_3/nlm_3$	70/30	30.39	0.7547
$tng3_3/nlm_3$	80/20	29.64	0.7692
$tng3_3/nlm_3$	90/10	29.04	0.7547

Table 8: Results $tng3_3/nlm3/generic$

Model	Weights	WER	F_1
$tng3_3/nlm_3/generic$	5/5/90	27.02	**0.9091**
$tng3_3/nlm_3/generic$	10/10/80	26.46	0.8889
$tng3_3/nlm_3/generic$	25/25/50	**25.08**	0.8333
$tng3_3/nlm_3/generic$	30/30/40	25.14	0.8333
$tng3_3/nlm_3/generic$	33/33/33	25.28	0.8511
$tng3_3/nlm_3/generic$	40/40/20	26.01	0.8333
$tng3_3/nlm_3/generic$	45/45/10	27.18	0.8000

6 Discussion

The results indicate that when interpolating a larger generic LM with a smaller domain-specific model, the search depth for each model is important to be aligned. Otherwise, the model with larger depth may take over as soon as the word length considered by the word search algorithm, leading to unintended results when interpolating multiple models. In our case, this gave disproportional weight to the domain-specific models and a higher WER, even in non-domain-specific sentences as can be noted when comparing the model $tng6_6/generic$ to $tng3_3/generic$ as shown in Appendix C.

Furthermore, the relatively high WER and number of false positive NEs of the NLM based model configurations, in comparison to the SLMs, suggests that the model was over-fitted to the training set. This is supported by the shape taken by the loss score in the training graph, Fig. 14, and validation graph, Fig. 15, which are exhibiting an L-shaped form. The L-shaped form is an indication that the model weights are being adjusted too much and over-fitted during the training process, in contrast to a more rounded shape which would indicate an appropriate learning rate. However, provided the long training time on the hardware available to the authors it was unfeasible within the scope of this project to both lower the learning rate and let the training process run for more epochs. This is however something that we believe could improve the results in terms of WER and NER for the NLM model individually as well as interpolated together with another domain-specific model and generic model, like the 25/25/50 and 5/5/90 models.

The worsening WER score seen in the SLM with an n-gram value of above 3 is likely due to the size of the corpus. As discussed in section 4.3.3, when increasing the length of the sentences not enough statistical information is collected in order to make a good prediction. In other words, there aren't enough repetitions of sentences of length 4 or above in order to create any meaningful statistical probability. With a larger corpus, the same sentences would be repeated enough to create a better prediction.

It is evident from the high number of false positives that both model types are indeed very biased towards finding the instruments in the voice recordings. It is not difficult to understand why when contemplating that all different in-

struments where replaced by a single tag. This tag then gets the accumulated probability of all different instruments it replaced.

However, all models were able to find all true positives except for one, and the question became one of achieving a lower WER and reducing false positives. It was observed that it may be possible to lower the domain-specific influence below the tested 10%, indicating that smaller domain-specific weights may be worth exploring. However, the 5/5/90 model was found to be the best performing model overall, even though it had a slightly worse WER than the 25/25/50 and 30/30/40 models.

The fact that the models when combined produced a better score in terms of NER false positives, compared to when running individually, shows that the different models have different strengths. When combined they essentially cancelled each others "mistakes" out.

The relation between the best performer in terms of WER and NER as shown in Table 8 illustrates that the different aspects of the system is not balanced. Increasing the influence of the two domain specific models to a combined influence of 50% yielded the best results in terms of WER but rendered the number of false positives so high that the weight distribution is almost unusable. However, when pushing the generic model forward in order to reduce the number of false positives a heavy toll was put on the WER score. This shows that the tools for calibrating the systems are blunt and would need to be reconsidered in an applied setting. In other words, it would be helpful to be able to tune the probability of a tagged entity by it's own and not only by influencing the whole transcription.

Ultimately, the choice of the best model may depend on the business case and the importance of reducing false positives versus accurate transcriptions. However, it is worth noting that the 5/5/90 model greatly reduced false positives when the NLM and SLMs were interpolated together. This indicates that these models have learned different ways of understanding the domain and can work better together to improve the accuracy of transcriptions.

Voice trading calls are typically not transcribed in a systematic way by respective organization as a part of trading, broking nor compliance. This means that quality data for model trading is not available for each independent institution that would benefit from a AVTS system. While the argument made about instrument and other name entity lists is that less data is needed for these models to be trained and used in the AVTS system, there is a question around the accuracy of the actual transcription and how heavily compliance personnel can rely on an automatic transcription versus listening to the actual call. Minor variances in phrasing and pronunciation can make a major difference to the meaning of said sentence. As a consequence, the question of whether a compliance officer's ethical responsibility to follow up alerts generated by the AVTS system can be elevated by dismissing alerts based on the automatic transcription is up for discussion. Should compliance officers be given the opportunity to do so at all or should the AVTS system simply supply hints of relevant discussions between brokers and clients, and only provide additional context to market events? The question is in part about how accurate transcriptions the system actually deliv-

ers. While most financial applications require absolute precision in the features they provide and hence should take serious caution in terms of conveying the output of difficult to explain methods, such as neural networks or approximation search heuristics, as an absolute truth to the end user, there is a case for the claim that voice trading surveillance may be subject to less strict precision requirements compared to other financial applications e.g. software that handles transfer of value like trading engines or banking systems. As long as the NER of instruments are accurate *enough* to provide compliance officers with a tool that can point them in the right direction in terms of linking electronic records of voice trades with likely candidates of phone calls where the trade was actually agreed it will go a long way in terms of helping financial market participants to comply with regulation. When it comes to displaying transcriptions made by the AVTS system however, it is important that the impression is not given that automatic transcriptions should be considered the absolute truth about what was being said between two humans and should be taken at face value. The less accurate transcriptions the system can *guarantee* the more logical it becomes to display meta data and analysis based on the transcription to the users, rather than the transcription itself.

7 Conclusion and Future Work

The study of interpolating smaller domain-specific models, in the financial domain, with a larger generic model proved successful, in relation to the criteria stated in section 4, as the WER was improved by interpolation and more than 75% of the NEs were found by all models. Furthermore, the AVTS system achieved decent accuracy by tuning the interpolation weights and the authors does consider the proof-of-concept successful provided that the model with the highest transcription accuracy, 25/25/50, only fell short of a 0.08 WER to the 25 WER criteria and that the model had an F_1-score above 0.8. Additionally, the authors think that the 5/5/90 model also deserves a mention in relation to the criteria with an F_1-score of 0.9091, even though it had a WER of 27.02. This means that the proposed AVTS system, and the method of instrument abstraction, is worthy of further evaluation and study.

While simple asset classes can be covered by the model by using the NE abstraction $< i >$ instrument tag, more complex asset classes like forwards and futures may benefit from more complex tag structures. Futures are contracts that oblige the holder to purchase a specific asset in the future at a specific price and date. These instruments are traditionally used to hedge against price fluctuation risks for industries producing or trading commodities. New contracts are created every day as dates mature with some interval e.g. three months, five months, one year etc. For this reason, new unique names are created every day. Instead of adding an infinite number of contract names to the vocabulary for the AVTS system to be able to transcribe them, one could divide future contract NE abstractions into parts e.g. $< i\ date >$ for dates, $< i\ price >$ for prices like settlement or strike price and $< i\ asset >$ for the underlying commodity

or other types of assets a contract is issued for. This way the AVTS models will be able to learn about complex contracts name structures while keeping configuration to a minimum.

Further analysis on the relationship between the number of configured instrument phonetic transcriptions and NER is warranted considering the low number of instruments included in the OMX30 index, Appendix F, which was used in the experiments compared to the full range of tradable instruments at an exchange. It is likely that the more phonetic transcriptions configured in the AVTS system, the worse the effect on WER and NE false positives due to the bias for instrument tag $< i >$ of the domain-specific models discussed in section 6.

Adapting the acoustic model will most certainly be of the utmost importance when implementing a production AVTS system. When the results for this report was produced, the recordings were done in a controlled environment with American English, the accent the AM is created for, and with very low background noise. In other words, the SNR was very high, with none, or very little, disturbance. In a production environment these conditions would most likely not be replicated and would likely do significant harm to the accuracy of the application. In order to help alleviate a lower SNR, the acoustic model can be adapted to any relevant accents or background noise.

8 Acknowledgements

The authors of this report would like to thank Scila AB that commissioned this Master's Thesis work and provided valuable feedback and the means to implement this project.

A special thank you to the native American English speaking students from Vanderbilt University in Nashville, Tennessee that provided us with the recordings used to evaluate the system.

The WER algorithm that has been used to measure the progress and success of the system has been developed by Martin Thoma, who was gracious enough to let us use if for purposes of this report [33].

References

[1] 111th Congress. Dodd–Frank Wall Street Reform and Consumer Protection Act. https://www.congress.gov/bill/111th-congress/house-bill/04173#major%20actions, 2010. [Online; accessed 3-Apr-2017].

[2] European Securities and Markets Authority. ESMA. https://www.esma.europa.eu/, 2017. [Online; accessed 7-Jan-2017].

[3] European Parliament, of the Council, and Commission Directives. Market Abuse Regulation. http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32014R0596&from=EN, 2014. [Online; accessed 3-Oct-2016].

[4] Dong Yu and Li Deng. *Automatic Speech Recognition: A Deep Learning Approach*. Springer, 2015.

[5] Paul Lamere, Philip Kwok, William Walker, Evandro B Gouvêa, Rita Singh, Bhiksha Raj, and Peter Wolf. Design of the cmu sphinx-4 decoder. In *INTERSPEECH*. Citeseer, 2003.

[6] Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, and Joe Woelfel. Sphinx-4: A flexible open source framework for speech recognition. 2004.

[7] CMU Sphinx 4 development team. Denoise Signal. http://cmusphinx.sourceforge.net/doc/sphinx4/edu/cmu/sphinx/frontend/denoise/Denoise.html, 2016. [Online; accessed 14-Nov-2016].

[8] Gerhard Doblinger. Computationally efficient speech enhancement by spectral minima tracking in subbands. *Power*, 1:2, 1995.

[9] Yuuki Tachioka, Shinji Watanabe, Jonathan Le Roux, and John R Hershey. Discriminative methods for noise robust speech recognition: A chime challenge benchmark. *Proc. CHiME-2013, Vancouver, Canada*, pages 19–24, 2013.

[10] Andrew Maas, Quoc V Le, Tyler M O'neil, Oriol Vinyals, Patrick Nguyen, and Andrew Y Ng. Recurrent neural networks for noise reduction in robust asr. 2012.

[11] Simone Cifani, Emanuele Principi, Cesare Rocchi, Stefano Squartini, and Francesco Piazza. A multichannel noise reduction front-end based on psychoacoustics for robust speech recognition in highly noisy environments. In *Hands-Free Speech Communication and Microphone Arrays, 2008. HSCMA 2008*, pages 172–175. IEEE, 2008.

[12] Hynek Hermansky. Perceptual linear predictive (plp) analysis of speech. *the Journal of the Acoustical Society of America*, 87(4):1738–1752, 1990.

[13] Sirko Molau, Michael Pitz, Ralf Schluter, and Hermann Ney. Computing mel-frequency cepstral coefficients on the power spectrum. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 1, pages 73–76. IEEE, 2001.

[14] Daniel Povey Oriol Vinyals, Suman Ravuri. Revisiting recurrent neural networks for robust asr. IEEE International Confrence on Acoustics, Speech, and Signal Processing (ICASSP), March 2012.

[15] Graham Upton and Ian Cook. *A dictionary of statistics 3e*. Oxford university press, 2014.

[16] CMU Sphinx 4 development team. Acoustic Model. `https://cmusphinx.github.io/wiki/acousticmodeltypes/`. [Online; accessed 2-Mar-2020].

[17] Wikipedia. Acoustic model, howpublished = "`https://en.wikipedia.org/wiki/Acoustic_model`", year = 2020.

[18] Dimitar D. Deliyski. Acoustic model and evaluation of pathological voice production. 1993.

[19] Dan Jurafsky and James H Martin. *Speech and language processing*. Pearson, 2014.

[20] Mehryar Mohri, Fernando Pereira, and Michael Riley. Speech recognition with weighted finite-state transducers. In *Springer Handbook of Speech Processing*, pages 559–584. Springer, 2008.

[21] Slava Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE transactions on acoustics, speech, and signal processing*, 35(3):400–401, 1987.

[22] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *journal of machine learning research*, 3(Feb):1137–1155, 2003.

[23] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

[24] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[25] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[26] Felix Gers. *Long short-term memory in recurrent neural networks*. PhD thesis, Universität Hannover, 2001.

[27] Xunying Liu, Xie Chen, Yongqiang Wang, Mark JF Gales, and Philip C Woodland. Two efficient lattice rescoring methods using recurrent neural network language models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(8):1438–1449, 2016.

[28] Holger Schwenk. Continuous space language models. *Computer Speech & Language*, 21(3):492–518, 2007.

[29] Dr. A.N.Cheeran Siddhant C. Joshi. Matlab based back-propagation neural network for automatic speech recognition. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 2014.

[30] RUSLAN SUVOROV JOHN LEVIS. Automatic speech recognition. 2012.

[31] Andrey Simanovsky Maksim Tkachenko. Named entity recognition: Exploring features. 5, 2012.

[32] Ali Mamat Alireza Mansouri, Lilly Suriani Affendey. Named entity recognition approaches. *IJCSNS International Journal of Computer Science and Network Security*, Febuary 2008.

[33] Martin Thoma. Word Error Rate Calculation. `https://martin-thoma.com/word-error-rate-calculation/`, 2013.

Acronyms

AM Acoustic Model. 4, 6, 7, 32

ASR Automatic Speech Recognition. 3–5, 7, 13–17, 19, 23

AVTS Automatic Voice Trade Surveillance. 3, 14–17, 19–21, 25, 30–32

CMU Carnegie Mellon University. 5, 6, 19, 22

D4J Deeplearning4j. 23

DFA Dodd–Frank Wall Street Reform and Consumer Protection Act. 3

DNN Deep Neural Network. 3, 12, 13

ESMA European Securities and Markets Authority. 3

FST Finite State Transducer. 7, 8, 12

G2P Grapheme to Phoneme. 18

HMM Hidden Markov Model. 5–7, 13, 17, 22

LM Language Model. 4, 6, 7, 12, 14, 16, 17, 19, 20, 22, 23, 29

LSTM Long Short-Term Memory. 12

MFCC Mel-Frequency Cepstral Coefficients. 5

NE Named Entity. 14, 15, 19, 20, 29, 31, 32

NER Named Entity Recognition. 3, 13, 14, 25, 27, 29–32

NLM Neural Language Model. 10, 23, 24, 27, 29, 30

OOV Out-of-Vocabulary. 17, 22

PLP Perceptual Linear Prediction. 5

PTM Phonetically tied model. 6

RNN Recurrent Neural Net. 10, 12, 23

SLM Stochastic Language Model. 7, 8, 22, 23, 27, 29, 30

SNR Signal-to-noise ratio. 5, 32

WER Word Error Rate. 4, 5, 14, 15, 25–32

Glossary

Market Data

Data generated by financial market market systems e.g. metadata of trades and orders such as price, volume and involved parties. 14

Markov Chain

A Markov Process is defined as a Markov Chain if all states has a common, finite number of outputs [15]. 7

Markov Process

A finite stochastic process where all states adhere to the Markov Property [15]. 5, 7, 37

Markov Property

The value of state s_t, at time t, is independent of all states prior to s_{t-1} [15]. 5, 37

Voice Trading

Industy term for trading performed over a call with a stock market broker. 3

A Statistic Language Model Results

Table 9: Word Error Rate *tng*

Model	Time	Rec 0	Rec 1	Rec 2	Rec 3	Average
$tng3_3$	0.82	**27.80**	24.54	30.98	**36.55**	**29.97**
$tng4_3$	0.79	28.96	24.54	30.98	**36.55**	30.26
$tng4_4$	0.90	29.73	23.05	**30.49**	38.28	30.39
$tng5_3$	0.79	**27.80**	24.54	30.98	**36.55**	**29.97**
$tng5_4$	0.91	29.73	23.05	**30.49**	38.28	30.39
$tng5_5$	1.01	30.50	**22.68**	30.98	38.28	30.61
$tng6_3$	0.81	**27.80**	24.54	30.98	**36.55**	**29.97**
$tng6_4$	0.94	30.89	23.05	**30.49**	38.28	30.68
$tng6_5$	1.09	30.50	**22.68**	30.98	38.28	30.61
$tng6_6$	1.17	30.50	**22.68**	31.22	38.28	30.67

Table 10: Named Entity Recognition *tng*

	Rec 0	Rec 1	Rec 2	Rec 3	Sum	F_1
$tng3_3$						
true positives	2	2	7	8	19	
false positives	1	3	5	3	12	0.7547
false negatives	0	0	1	0	1	
$tng4_3$						
true positives	2	2	7	8	19	
false positives	1	3	5	3	12	0.7547
false negatives	0	0	1	0	1	
$tng4_4$						
true positives	2	2	7	8	19	
false positives	1	4	6	3	14	0.7273
false negatives	0	0	1	0	1	
$tng5_3$						
true positives	2	2	7	8	19	
false positives	1	3	5	3	12	0.7547
false negatives	0	0	1	0	1	
$tng5_4$						
true positives	2	2	7	8	19	
false positives	1	4	6	3	14	0.7273
false negatives	0	0	1	0	1	
$tng5_5$						
true positives	2	2	7	8	19	
false positives	1	4	6	3	14	0.7273
false negatives	0	0	1	0	1	
$tng6_3$						
true positives	2	2	7	8	19	
false positives	1	3	5	3	12	0.7547
false negatives	0	0	1	0	1	
$tng6_4$						
true positives	2	2	7	8	19	
false positives	1	4	6	3	14	0.7273
false negatives	0	0	1	0	1	
$tng6_5$						
true positives	2	2	7	8	19	
false positives	1	4	6	3	14	0.7273
false negatives	0	0	1	0	1	
$tng6_6$						
true positives	2	2	7	8	19	
false positives	1	4	6	3	14	0.7273
false negatives	0	0	1	0	1	

B Neural Languagel Model Results

Table 11: Word Error Rate *nlm*

Model	Time	Rec 0	Rec 1	Rec 2	Rec 3	Average
nlm_3	11.89	44.40	**39.78**	37.07	**42.76**	41.00
nlm_4	23.80	42.86	41.26	36.59	43.10	40.95
nlm_5	30.34	**42.08**	41.64	36.10	43.79	**40.90**
nlm_6	34.81	42.86	41.64	**35.61**	44.48	41.15

Table 12: Named Entity Recognition *nlm*

	Rec 0	Rec 1	Rec 2	Rec 3	Sum	F_1
nlm_3						
true positives	2	2	7	8	19	
false positives	4	7	3	3	17	0.6897
false negatives	0	0	1	0	1	
nlm_4						
true positives	2	2	7	8	19	
false positives	3	8	3	2	16	0.7018
false negatives	0	0	1	0	1	
nlm_5						
true positives	2	2	7	8	19	
false positives	2	8	3	2	15	0.7143
false negatives	0	0	1	0	1	
nlm_6						
true positives	2	2	7	8	19	
false positives	2	8	3	2	15	0.7143
false negatives	0	0	1	0	1	

C Two Models Interpolation Result

Table 13: Interpolated Model Word Error Rate

Model	Weights	Time	Rec 0	Rec 1	Rec 2	Rec 3	Average
$tng3_3/generic$	10/90	1.08	26.64	21.19	26.34	36.90	27.77
$tng3_3/generic$	20/80	1.80	25.48	20.07	25.12	36.55	26.81
$tng3_3/generic$	30/70	1.20	26.25	18.96	25.37	35.52	26.52
$tng3_3/generic$	40/60	0.74	25.48	18.96	25.37	**34.48**	26.07
$tng3_3/generic$	50/50	0.88	25.10	**18.22**	**24.88**	34.83	**25.75**
$tng3_3/generic$	60/40	0.70	24.71	**18.22**	25.37	35.17	25.87
$tng3_3/generic$	70/30	0.74	24.71	18.59	25.85	35.17	26.08
$tng3_3/generic$	80/20	1.00	**23.94**	18.59	25.85	35.52	25.97
$tng3_3/generic$	90/10	0.72	25.48	19.70	26.10	**34.48**	26.44
$tng6_6/generic$	10/90	0.98	31.66	23.79	29.76	37.93	30.78
$tng6_6/generic$	20/80	0.94	31.27	24.16	30.00	37.59	30.76
$tng6_6/generic$	30/70	0.94	31.27	24.16	30.24	37.59	30.82
$tng6_6/generic$	40/60	1.00	31.27	23.79	30.00	37.59	30.66
$tng6_6/generic$	50/50	1.00	31.27	23.42	29.02	37.24	30.24
$tng6_6/generic$	60/40	0.96	31.27	22.68	29.51	37.24	30.18
$tng6_6/generic$	70/30	0.93	31.27	23.05	29.27	37.93	30.38
$tng6_6/generic$	80/20	0.91	31.66	22.30	29.27	38.28	30.38
$tng6_6/generic$	90/10	0.91	31.27	22.30	30.24	38.28	30.52

Table 14: Interpolated Model Word Error Rate

Model	Weights	Time	Rec 0	Rec 1	Rec 2	Rec 3	Average
$nlm_3/generic$	10/90	9.21	32.43	24.91	28.78	37.59	30.93
$nlm_3/generic$	20/80	9.28	31.66	26.02	27.07	36.90	30.41
$nlm_3/generic$	30/70	9.39	29.73	26.02	27.07	36.55	29.84
$nlm_3/generic$	40/60	9.50	30.12	26.02	26.83	35.52	29.62
$nlm_3/generic$	50/50	9.57	30.12	25.65	27.32	35.86	29.74
$nlm_3/generic$	60/40	9.61	30.12	26.77	27.32	35.52	29.93
$nlm_3/generic$	70/30	9.66	30.89	26.77	28.54	36.21	30.6
$nlm_3/generic$	80/20	9.74	32.43	28.25	30.24	37.24	32.04
$nlm_3/generic$	90/10	9.81	34.75	30.11	31.22	37.59	33.42
$nlm_5/generic$	10/90	27.34	40.93	41.26	34.39	43.45	40.01
$nlm_5/generic$	20/80	48.28	41.70	40.52	34.88	43.10	40.05
$nlm_5/generic$	30/70	26.73	41.70	40.52	34.63	43.10	39.99
$nlm_5/generic$	70/30	26.70	40.93	40.52	34.63	43.10	39.80
$nlm_5/generic$	80/20	26.60	40.93	40.89	36.10	43.10	40.25
$nlm_5/generic$	90/10	26.57	40.93	41.26	36.83	43.10	40.53

Table 15: Interpolated Model Word Error Rate

Model	Weights	Time	Rec 0	Rec 1	Rec 2	Rec 3	Average
$tng3_3/nlm_3$	10/90	12.09	35.91	30.86	33.41	40.00	35.04
$tng3_3/nlm_3$	20/80	11.66	35.52	27.88	31.71	39.66	33.69
$tng3_3/nlm_3$	30/70	11.59	34.75	25.28	31.22	39.31	32.64
$tng3_3/nlm_3$	40/60	11.41	33.98	24.16	30.49	37.59	31.55
$tng3_3/nlm_3$	50/50	11.29	32.05	24.54	30.00	37.93	31.13
$tng3_3/nlm_3$	60/40	11.18	31.66	23.42	30.00	37.59	30.67
$tng3_3/nlm_3$	70/30	11.03	30.89	23.79	30.00	36.90	30.39
$tng3_3/nlm_3$	80/20	10.82	28.19	24.16	30.00	36.21	29.64
$tng3_3/nlm_3$	90/10	10.82	28.19	23.05	29.76	35.17	29.04
$tng6_5/nlm_5$	10/90	30.02	35.14	33.46	33.17	37.59	34.84
$tng6_5/nlm_5$	20/80	29.81	33.59	29.37	32.93	36.90	33.20
$tng6_5/nlm_5$	30/70	83.11	32.82	26.02	31.95	38.28	32.27
$tng6_5/nlm_5$	70/30	26.62	28.96	25.28	31.46	36.21	30.48
$tng6_5/nlm_5$	80/20	26.15	27.80	23.79	31.46	37.24	30.07
$tng6_5/nlm_5$	90/10	25.46	29.34	22.30	31.22	37.24	30.03

Table 16: Named Entity Recognition $tng3_3/generic$

	Weights	Rec 0	Rec 1	Rec 2	Rec 3	Sum	F_1
	10/90						
true positives		2	2	7	8	19	
false positives		0	3	2	4	9	0.8000
false negatives		0	0	1	0	1	
	20/80						
true positives		2	2	7	8	19	
false positives		0	3	2	4	9	0.8000
false negatives		0	0	1	0	1	
	30/70						
true positives		2	2	7	8	19	
false positives		0	3	2	4	9	0.8000
false negatives		0	0	1	0	1	
	40/60						
true positives		2	2	7	8	19	
false positives		0	3	3	4	9	0.8000
false negatives		0	0	1	0	1	
	50/60						
true positives		2	2	7	8	19	
false positives		0	3	3	4	9	0.8000
false negatives		0	0	1	0	1	
	60/40						
true positives		2	2	7	8	19	
false positives		0	4	3	4	9	0.8000
false negatives		0	0	1	0	1	
	70/30						
true positives		2	2	7	8	19	
false positives		0	4	3	4	9	0.8000
false negatives		0	0	1	0	1	
	80/20						
true positives		2	2	7	8	19	
false positives		0	4	3	4	9	0.8000
false negatives		0	0	1	0	1	
	90/10						
true positives		2	2	7	8	19	
false positives		1	4	3	4	10	0.7843
false negatives		0	0	1	0	1	

Table 17: Named Entity Recognition $tng6_6/generic$

	Weights	Rec 0	Rec 1	Rec 2	Rec 3	Sum	F_1
	10/90						
true positives		2	2	7	8	19	
false positives		2	4	4	4	14	0.7273
false negatives		0	0	1	0	1	
	20/80						
true positives		2	2	7	8	19	
false positives		2	4	4	4	14	0.7273
false negatives		0	0	1	0	1	
	30/70						
true positives		2	2	7	8	19	
false positives		1	4	4	3	12	0.7547
false negatives		0	0	1	0	1	
	40/60						
true positives		2	2	7	8	19	
false positives		1	4	4	3	12	0.75471
false negatives		0	0	1	0	1	
	50/50						
true positives		2	2	7	8	19	
false positives		1	4	4	4	13	0.7407
false negatives		0	0	1	0	1	
	60/40						
true positives		2	2	7	8	19	
false positives		1	4	4	4	13	0.7407
false negatives		0	0	1	0	1	
	70/30						
true positives		2	2	7	8	19	
false positives		1	4	4	4	13	0.7407
false negatives		0	0	1	0	1	
	80/20						
true positives		2	2	7	8	19	
false positives		1	4	4	4	13	0.7407
false negatives		0	0	1	0	1	
	90/10						
true positives		2	2	7	8	19	
false positives		1	4	6	4	15	0.7143
false negatives		0	0	1	0	1	

Table 18: Named Entity Recognition $nlm_3/generic$

	Weights	Rec 0	Rec 1	Rec 2	Rec 3	Sum	F_1
	10/90						
true positives		2	2	7	8	19	
false positives		1	6	3	3	13	0.7407
false negatives		0	0	1	0	1	
	20/80						
true positives		2	2	7	8	19	
false positives		1	7	3	3	14	0.7273
false negatives		0	0	1	0	1	
	30/70						
true positives		2	2	7	8	19	
false positives		0	7	3	3	13	0.7407
false negatives		0	0	1	0	1	
	40/60						
true positives		2	2	7	8	19	
false positives		0	7	3	3	13	0.7407
false negatives		0	0	1	0	1	
	50/50						
true positives		2	2	7	8	19	
false positives		0	7	2	3	12	0.7547
false negatives		0	0	1	0	1	
	60/40						
true positives		2	2	7	8	19	
false positives		0	6	2	3	11	0.7692
false negatives		0	0	1	0	1	
	70/30						
true positives		2	2	7	8	19	
false positives		1	6	2	3	12	0.7547
false negatives		0	0	1	0	1	
	80/20						
true positives		2	2	7	8	19	
false positives		1	6	2	3	12	0.7547
false negatives		0	0	1	0	1	
	90/10						
true positives		2	2	7	8	19	
false positives		1	6	2	3	12	0.7547
false negatives		0	0	1	0	1	

Table 19: Named Entity Recognition $nlm_5/generic$

	Weights	Rec 0	Rec 1	Rec 2	Rec 3	Sum	F_1
	10/90						
true positives		2	2	7	8	19	
false positives		3	8	4	2	17	0.6897
false negatives		0	0	1	0	1	
	20/80						
true positives		2	2	7	8	19	
false positives		3	8	4	2	17	0.6897
false negatives		0	0	1	0	1	
	30/70						
true positives		2	2	7	8	19	
false positives		3	8	4	2	17	0.6897
false negatives		0	0	1	0	1	
	70/30						
true positives		2	2	7	8	19	
false positives		2	8	4	2	16	0.7018
false negatives		0	0	1	0	1	
	80/20						
true positives		2	2	7	8	19	
false positives		2	8	4	2	16	0.7018
false negatives		0	0	1	0	1	
	90/10						
true positives		2	2	7	8	19	
false positives		2	8	3	2	15	0.7018
false negatives		0	0	1	0	1	

Table 20: Named Entity Recognition $tng3_3/nlm_3$

	Weights	Rec 0	Rec 1	Rec 2	Rec 3	Sum	F_1
	10/90						
true positives		2	2	7	8	19	
false positives		3	4	3	3	13	0.7407
false negatives		0	0	1	0	1	
	20/80						
true positives		2	2	7	8	19	
false positives		2	3	2	3	10	0.7843
false negatives		0	0	1	0	1	
	30/70						
true positives		2	2	7	8	19	
false positives		2	3	2	3	10	0.7843
false negatives		0	0	1	0	1	
	40/60						
true positives		2	2	7	8	19	
false positives		2	3	2	3	10	0.7843
false negatives		0	0	1	0	1	
	50/50						
true positives		2	2	7	8	19	
false positives		2	3	3	3	11	0.7692
false negatives		0	0	1	0	1	
	60/40						
true positives		2	2	7	8	19	
false positives		1	3	4	3	11	0.7692
false negatives		0	0	1	0	1	
	70/30						
true positives		2	2	7	8	19	
false positives		1	4	4	3	12	0.7547
false negatives		0	0	1	0	1	
	80/20						
true positives		2	2	7	8	19	
false positives		1	3	4	3	11	0.7692
false negatives		0	0	1	0	1	
	90/10						
true positives		2	2	7	8	19	
false positives		1	3	5	3	12	0.7547
false negatives		0	0	1	0	1	

Table 21: Named Entity Recognition $tng6_5/nlm_5$

	Weights	Rec 0	Rec 1	Rec 2	Rec 3	Sum	F_1
	10/90						
true positives		2	2	7	8	19	
false positives		2	5	3	3	13	0.7407
false negatives		0	0	1	0	1	
	20/80						
true positives		2	2	7	8	19	
false positives		2	4	3	3	12	0.7547
false negatives		0	0	1	0	1	
	30/70						
true positives		2	2	7	8	19	
false positives		2	4	3	3	12	0.7547
false negatives		0	0	1	0	1	
	70/30						
true positives		2	2	7	8	19	
false positives		1	4	5	3	13	0.7407
false negatives		0	0	1	0	1	
	80/20						
true positives		2	2	7	8	19	
false positives		1	3	5	3	12	0.7547
false negatives		0	0	1	0	1	
	90/10						
true positives		2	2	7	8	19	
false positives		1	4	6	3	14	0.7273
false negatives		0	0	1	0	1	

D Three Models Interpolation Results

Table 22: Word Error Rate $tng3_3/nlm3/generic$

Model	Weights	Time	Rec 0	Rec 1	Rec 2	Rec 3	Average
$tng3_3/nlm_3/generic$	5/5/90	8.46	26.25	20.45	25.85	35.52	27.02
$tng3_3/nlm_3/generic$	10/10/80	8.84	25.48	20.82	26.10	33.45	26.46
$tng3_3/nlm_3/generic$	25/25/50	9.62	**22.78**	**17.84**	26.59	**33.10**	**25.08**
$tng3_3/nlm_3/generic$	30/30/40	9.88	**22.78**	18.59	26.10	**33.10**	25.14
$tng3_3/nlm_3/generic$	33/33/33	9.86	24.32	18.59	**25.12**	**33.10**	25.28
$tng3_3/nlm_3/generic$	40/40/20	10.03	25.87	18.96	26.10	**33.10**	26.01
$tng3_3/nlm_3/generic$	45/45/10	10.22	26.64	20.82	26.10	35.17	27.18

Table 23: Named Entity Recognition $tng3_3/nlm_3/generic$

Model	Weights	Rec 0	Rec 1	Rec 2	Rec 3	Sum	F_1
	5/5/90/						
true positives		2	2	7	8	19	
false positives		0	2	0	1	3	0.9091
false negatives		0	0	1	0	1	
	10/10/80/						
true positives		2	2	7	8	19	
false positives		0	3	0	1	4	0.8889
false negatives		0	0	1	0	1	
	25/25/50/						
true positives		2	2	7	8	19	
false positives		0	3	2	2	7	0.8333
false negatives		0	0	1	0	1	
	30/30/40/						
true positives		2	2	7	8	19	
false positives		0	3	2	2	7	0.8333
false negatives		0	0	1	0	1	
	33/33/33/						
true positives		2	2	7	8	19	
false positives		0	2	2	2	6	0.8511
false negatives		0	0	1	0	1	
	40/40/20/						
true positives		2	2	7	8	19	
false positives		0	2	2	3	7	0.8333
false negatives		0	0	1	0	1	
	45/45/10/						
true positives		2	2	7	8	19	
false positives		1	2	3	3	9	0.8000
false negatives		0	0	1	0	1	

E Pronunciation Dictionary

Phoneme	Example	Translation
AA	odd	AA D
AE	at	AE T
AH	hut	HH AH T
AO	ought	AO T
AW	cow	K AW
AY	hide	HH AY D
B	be	B IY
CH	cheese	CH IY Z
D	dee	D IY
DH	thee	DH IY
EH	Ed	EH D
ER	hurt	HH ER T
EY	ate	EY T
F	fee	F IY
G	green	G R IY N
HH	he	HH IY
IH	it	IH T
IY	eat	IY T
JH	gee	JH IY
K	key	K IY
L	lee	L IY
M	me	M IY
N	knee	N IY
NG	ping	P IH NG
OW	oat	OW T
OY	toy	T OY
P	pee	P IY
R	read	R IY D
S	sea	S IY
SH	she	SH IY
T	tea	T IY
TH	theta	TH EY T AH
UH	hood	HH UH D
UW	two	T UW
V	vee	V IY
W	we	W IY
Y	yield	Y IY L D
Z	zee	Z IY
ZH	seizure	S IY ZH ER

Table sourced from http://www.speech.cs.cmu.edu/cgi-bin/cmudict, related dictonary source files can be found at https://github.com/cmusphinx/cmudict.

F Instrument Dictionary

Instrument	Phonetics
ABB	EY B IY B IY
ABB_Ltd	EY B IY B IY EH L T IY D IY
ALFA	EY EH L EH F EY
ALFA	AE L F AH
Alfa_Laval	AE L F AH L AA V AA L
ALIV_SDB	EY EH L AY V
ALIV_SDB	EY EH L AY V IY S D IY B IY
Autoliv	AO T OW L IH V
Autoliv_SDB	AO T OW L IH V EH S D IY B IY
ASSA_Abloy	AE S AH AH B L OY
ASSA	AE S AH
ASSA	EY EH S EH S EY
ASSA_B	EY EH S EH S EY B IY
ASSA_B	AE S AH B IY
ASSA_Abloy_B	AE S AH AH B L OY B IY
Atlas_Copco	AE T L AH S K AA P K OW
Atlas_Copco	AE T L AH Z K AA P K OW
Atlas_Copco	AE T L AH S K OW P K OW
Atlas_Copco	AE T L AH Z K OW P K OW
ATCO	AE K T OW
ATCO	EY T IY S IY OW
ATCO_A	AE K T OW EY
ATCO_A	EY T IY S IY OW EY
Atlas_Copco_A	AE T L AH S K AA P K OW EY
Atlas_Copco_A	AE T L AH Z K AA P K OW EY
Atlas_Copco_A	AE T L AH S K OW P K OW EY
Atlas_Copco_A	AE T L AH Z K OW P K OW EY
Atlas_Copco_A	AE T L AH S K AA P K OW B IY
Atlas_Copco_A	AE T L AH Z K AA P K OW B IY
Atlas_Copco_A	AE T L AH S K OW P K OW B IY
Atlas_Copco_A	AE T L AH Z K OW P K OW B IY
AstraZeneca	AE S T R AH Z EH N AH K AH
AstraZeneca	AE S T R AH Z EH N EH K AH
Boliden	B OW L IH D AH N
Boliden	B OW L IH D EH N
Boliden	B OW L IY D AH N
Boliden	B OW L IY D EH N
BOL	B OW EH L
BOL	B AA L
BOL	B OW L

Instrument	Phonetics
Electrolux	IH L EH K T R AH L AH K S
Electrolux	EH L EH K T R AH L AH K S
Electrolux	IH L EH K T R OW L AH K S
Electrolux	EH L EH K T R OW L AH K S
ELUX	IH L AH K S
ELUX	IY EH L Y UW EH K S
ELUX_B	IH L AH K S B IY
ELUX_B	IY EH L Y UW EH K S B IY
Electrolux_B	IH L EH K T R AH L AH K S B IY
Electrolux_B	EH L EH K T R AH L AH K S B IY
Electrolux_B	IH L EH K T R OW L AH K S B IY
Electrolux_B	EH L EH K T R OW L AH K S B IY
Ericsson	EH R IH K S AH N
Ericsson	EH R IH K S AO N
Ericsson	EH R IH K S AH N
ERIC	EH R IH K
ERIC	IY AA R AY S IY
ERIC_B	EH R IH K B IY
ERIC_B	IY AA R AY S IY B IY
Ericsson_B	EH R IH K S AH N B IY
Ericsson_B	EH R IH K S AO N B IY
Fingerprint	F IH NG G ER P R IH N T
Fingerprint_Cards	F IH NG G ER P R IH N T K AA R D Z
Fingerprint_Cards_B	F IH NG G ER P R IH N T K AA R D Z B IY
FING	F IH NG
FING	EH F AY EH N JH IY
FING_B	F IH NG B IY
FING_B	EH F AY EH N JH IY B IY
Getinge	G EH T IH NG EH
Getinge	G EH T IH NG
Getinge	JH EH T IH NG EH
Getinge_B	G EH T IH NG EH B IY
GETI_B	G EH T IH B IY
GETI_B	G EH T AY B IY
Hennes_&_Mauritz	EY CH AH N D EH M
Hennes_&_Mauritz	HH EH N IY Z AH N D M AO R IH T S
Hennes_&_Mauritz_B	EY CH AH N D EH M B IY
Hennes_&_Mauritz_B	HH EH N IY Z AH N D M AO R IH T S B IY
Investor	IH N V EH S T AO R
Investor	IH N V EH S T ER
Investor_B	IH N V EH S T AO R B IY
Investor_B	IH N V EH S T ER B IY

Instrument	Phonetics
Kinnevik	K IH N IH V IH K
Kinnevik	K IH N EH V IH K
Kinnevik	CH IH N EH V IY K
Lundin_Petroleum	L AH N D IH N P AH T R OW L IY AH M
Lundin_Petroleum	L UH N D IH N P AH T R OW L IY AH M
Lundin_Petroleum	L UW N D IH N P AH T R OW L IY AH M
Lundin_Petroleum	L AH N D IH N P EH T R OW L IY AH M
Lundin_Petroleum	L UH N D IH N P EH T R OW L IY AH M
Lundin_Petroleum	L UW N D IH N P EH T R OW L IY AH M
Nordea	N AO R D IY AH
Nordea_Bank	N AO R D IY AH B AE NG K
Sandvik	S AE N D V IH K
Sandvik	S AE N D V IY K
SCA	EH S S IY EY
SCA_B	EH S S IY EY B IY
SEB	EH S IY B IY
SEB_A	EH S IY B IY EY
Securitas	S IH K Y UH R AH T AH Z
Securitas_B	S IH K Y UH R AH T AH Z B IY
Sv_Handelsbanken	HH AE N D AH L S B AE NG K AH N
Sv_Handelsbanken_A	HH AE N D AH L S B AE NG K AH N EY
Skanska	S K AE N S K AH
Skanska_B	S K AE N S K AH B IY
SKF_B	EH S K EY EH F
SKF_B	EH S K EY EH F B IY
SSAB	EH S EH S EY B IY
SSAB_A	EH S EH S EY B IY EY
Swedbank	S W EH D B AH NG K
Swedbank_A	S W EH D B AH NG K EY
Swedish_Match	S W IY D IH SH M AE CH
Tele2	T EH L AH T UW
Tele2	T EH L EH T UW
Tele2	T EH L IY T UW
Tele2_B	T EH L AH T UW B IY
Tele2_B	T EH L EH T UW B IY
TELIA	T EH L IY AH
TELIAs	T EH L IY AH S
TELIAs	T EH L IY AH Z
Telia_Company	T EH L IY AH K AH M P AH N IY
Volvo	V OW L V OW
Volvo_B	V OW L V OW B IY

G Number Dictionary

Number	Phoneme
<n>one</n>	W AH N
<n>one</n>	HH W AH N
<n>one's</n>	W AH N Z
<n>two</n>	T UW
<n>two's</n>	T UW Z
<n>three</n>	TH R IY
<n>three's</n>	TH R IY Z
<n>four</n>	F AO R
<n>four's</n>	F AO R Z
<n>five</n>	F AY V
<n>five</n>	F AY F
<n>five's</n>	F AY V Z
<n>six</n>	S IH K S
<n>six's</n>	S IH K S IH Z
<n>seven</n>	S EH V AH N
<n>seven's</n>	S EH V AH N Z
<n>eight</n>	EY T
<n>eight's</n>	EY T S
<n>nine</n>	N AY N
<n>nine's</n>	N AY N Z
<n>ten</n>	T EH N
<n>ten's</n>	T EH N Z
<n>eleven</n>	IH L EH V AH N
<n>eleven</n>	IY L EH V AH N
<n>eleven's</n>	IH L EH V AH N Z
<n>eleven's</n>	IY L EH V AH N Z
<n>twelve</n>	T W EH L V
<n>thirteen</n>	TH ER T IY N
<n>thirteen's</n>	TH ER T IY N Z
<n>fourteen</n>	F AO R T IY N
<n>fourteen's</n>	F AO R T IY N Z
<n>fifteen</n>	F IH F T IY N
<n>fifteen's</n>	F IH F T IY N Z
<n>sixteen</n>	S IH K S T IY N
<n>sixteen's</n>	S IH K S T IY N Z
<n>seventeen</n>	S EH V AH N T IY N
<n>seventeen's</n>	S EH V AH N T IY N Z
<n>eighteen</n>	EY T IY N
<n>eighteen's</n>	EY T IY N Z
<n>nineteen</n>	N AY N T IY N
<n>nineteen's</n>	N AY N T IY N

Number	Phoneme
<n>twenty</n>	T W EH N T IY
<n>twenty</n>	T W EH N IY
<n>twenty's</n>	T W EH N T IY Z
<n>twenty's</n>	T W EH N IY Z
<n>thirty</n>	TH ER D IY
<n>thirty</n>	TH ER T IY
<n>thirty's</n>	TH ER D IY Z
<n>thirty's</n>	TH ER T IY Z
<n>forty</n>	F AO R T IY
<n>forty</n>	F AO R D IY
<n>forty's</n>	F AO R T IY Z
<n>forty's</n>	F AO R D IY Z
<n>fifty</n>	F IH F T IY
<n>fifty's</n>	F IH F T IY Z
<n>sixty</n>	S IH K S T IY
<n>sixty's</n>	S IH K S T IY Z
<n>seventy</n>	S EH V AH N T IY
<n>seventy's</n>	S EH V AH N T IY Z
<n>eighty</n>	EY T IY
<n>eighty's</n>	EY T IY Z
<n>ninety</n>	N AY N T IY
<n>ninety's</n>	N AY N T IY Z
<n>hundred</n>	HH AH N D R AH D
<n>hundred</n>	HH AH N D R IH D
<n>hundred</n>	HH AH N ER D
<n>hundred</n>	HH AH N D ER D
<n>hundreds</n>	HH AH N D R IH D Z
<n>hundreds</n>	HH AH N D ER D Z
<n>hundreds</n>	HH AH N ER D Z
<n>hundreds</n>	HH AH N D R AH D Z
<n>thousand</n>	TH AW Z AH N D
<n>thousand</n>	TH AW Z AH N
<n>thousands</n>	TH AW Z AH N D Z
<n>thousands</n>	TH AW Z AH N Z
<n>million</n>	M IH L Y AH N
<n>millions</n>	M IH L Y AH N Z
<n>billion</n>	B IH L Y AH N
<n>billions</n>	B IH L Y AH N Z
<n>milliard</n>	M IH L Y AH R D
<n>trillion</n>	T R IH L Y AH N
<n>trillions</n>	T R IH L Y AH N Z
<n>market</n>	M AA R K AH T
<n>market</n>	M AA R K IH T