Universal Tagger

A thesis presented
by

Long Thanh Duong

to
The Department of Computing and Information System
in total fulfillment of the requirements
for the degree of
Master by Course Work

The University of Melbourne
Melbourne, Australia
June 2013

Declaration

This is to certify that:

(i) this thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person where due reference is not made in the text;

(ii) where necessary I have received clearance for this research from the University's Ethics Committee and have submitted all required data to the Department;

(iii) the thesis is fewer than 20,000 words in length, exclusive of tables, maps, bibliographies and appendices.

Signed: _____ Date: _____

Thesis advisor(s)
A/Prof. Steven Bird
Dr. Paul Cook

Author
Long Thanh Duong

Universal Tagger

Abstract

Part-of-speech (POS) tagging is one of the most basic and crucial tasks in Natural Language Processing (NLP). Supervised POS taggers perform well on many resource-rich languages i.e. English, French, Portuguese etc, where manually annotated data is available. However, it is impossible to use a supervised approach for the vast number of resource-poor languages. In this thesis, we apply a multilingual unsupervised method for building taggers for resource-poor languages base additionally on parallel data (Universal Tagger), that is, we use parallel data as the bridge to transfer tag information from resource-rich to resource-poor languages. On average, our tagger performs on par with the state of the art on the same test set of eight languages. However, we use less data and a less sophisticated method which also results in significant difference in speed. In an effort to further improve performance, we investigate the choice of source language. We found that English is rarely the best source language. We successfully built a model that can predict the best source language only based on monolingual data. However, even better predictions can be made if we additionally use parallel data. Finally, we show that, if multiple source languages are available, it is possible to get further improvement by incorporating them.

Contents

List of Figures

List of Tables

Citations to Previously Published Work

Large portions of chapter 3 have appeared in the following paper:

> **Long Duong**, Paul Cook, Steven Bird, and Pavel Pecina. 2013. Simpler unsupervised POS tagging with bilingual projections. In *Proceedings of the main conference 51th Annual Meeting of the Association for Computational Linguistics (ACL 2013)*. Association for Computational Linguistics, Sofia, Bulgaria

Acknowledgments

My family is the ultimate motivation for me. I want to thank my mom, my dad and my sister who always encourage me to finish this work. I thank my uncle and my aunt for helping me with the daily life. Her cooking is something I'm looking forward to everyday back from University. Frankly speaking, she saved me a lot of time for cooking, washing and all. I thank Siva Reddy for countless useful discussions and suggestions. I also would like to thank Spandana Gella for all the encouragement she generously gave. Thank you Tiger, Yahya, Min, Yousef, Mahtab for making my time in Melbourne enjoyable. Thank Duong Ly for a great deal of motivation and help from very first moment I was in Melbourne. Finally, I want to thank Kleine Mausi who always stands by my side, encourages me, motivates me and inspires me.

Dedicated to youse all.

Chapter 1

Introduction

1.1 Part-of-speech Tagging

A Part-of-speech (POS) is the syntactic category of a lexical item (word). Lexical items which share the same POS are believed to have similar syntactic and morphological behavior. Common POS include Verb, Noun, Adjective, Pronoun, Adverb, Preposition and so forth. POS information is used to disambiguate different syntactic categories. For example, in the sentence "*We can can a can*", the word "*can*" belongs to different categories: (1) a modal verb, ie, *somebody can do something*; (2) a verb which is making a bottle; (3) a noun which refers to a container. A POS tagger is a system for automatically determining the POS tag for a given text, and it should be able to distinguish syntactic categories by assigning tags to words. The above sentence can be tagged as "$We_N\ can_{MD}\ can_V\ a_{Det}\ can_N$" where N is noun, MD is modal verb, V is verb, Det is determiner. Another example is the word "*running*", the common interpretation is a verb i.e. "*he is running*". However, it could be an adjective i.e. "*running shoes*" or noun i.e. "*running is good for health*". Therefore, sometimes POS tagging is also referred as choosing the right sense for the right context. Nevertheless, it is worth noting that when the same word-form has different tags, it will differ in meaning, but the reverse statement is not always true. For example, the word "*bank*" might refer to (1) the place where people conduct monetary transaction (2) the place near the river, but both meanings have the same "*Noun*" POS.

POS tagging is one of the most basic operations of computational linguistic. Since it helps to disambiguate syntactic categories (and possibly senses), POS are regularly used in various natural language processing (NLP) tasks such as parsing, sentence classifying, word sense disambiguation and so-forth.

1.2 Tag set

Tag set is the list of possible tags that a lexical item (word) can have. Normally, the tag set is different across languages. For English, the well-known tag set is Penn Treebank (Marcus *et al.* 1993) tag set which contains 48 tags; for more information please see Appendix A. The tag set is used to disambiguate morphology and syntax of lexical items. Therefore, the number of tags denotes the morphological complexity of a language. For example, there are 67 classes in the Prague Dependency Tree Bank (Hajič *et al.* 2000) which shows the greater morphological complexity of Czech language compared with English.

1.3 Evaluation

The most straightforward evaluation for POS tagging is *per-token* accuracy. Firstly, sentence is tokenized, which will separate lexicon items from each other. Normally, space is used as the delimiter and period "." indicates a sentence boundary. However, there are cases where this will not work, for example, the sentence *"Mr. Peter ate a banana"* should be tokenized as *(Mr.) (Peter) (ate) (a) (banana)* but not *(Mr) (.) (Peter) (ate) (a) (banana)*. Second, the whole sentence is tagged. The tag of each token is then compared with the gold standard test data which is manually annotated by a linguist. The percentage of correctly tagged for tokens is known as the per-token accuracy. This measure is widely applied and has become the default evaluation. Supervised POS taggers achieve as high as 97 % per-token accuracy for English (Toutanova *et al.* 2003a) and for many other languages.

People might argue that this per-token accuracy is not meaningful because it takes into account punctuation marks and tokens that are not ambiguous. Therefore, another evaluation metric is *ambiguous-token* accuracy which only counts tokens that have more than one possible tag. The list of possible tags for each word can be acquired using dictionary or large manually annotated data. To the best of our knowledge, the best system (Yoshida *et al.* 2007) reports ambiguous-token accuracy around 87%.

Other evaluation is *per-sentence* accuracy. That is, accuracy is calculated on the number of sentences for which all tokens are correctly tagged. This measure is more meaningful for tasks such as dependency-parsing where a single mistake in POS tagging might lead to completely wrong parsed sentence. The current good taggers report per-sentence accuracy around 55-57 % (Manning 2011). In this thesis, we employ **per-token accuracy** for comparison purposes, thus, when we report accuracy of a tagger, we implicitly mean per-token accuracy.

1.4 Traditional approach

The traditional approach to POS tagging builds a separate tagger for each target language. It does not take into consideration the relationship between languages. The supervised method which employs supervised machine learning algorithms is usually used. For each language they collect and build a large amount of manually annotated data and train a supervised POS tagger on it. The supervised style for the traditional approach has achieved very high tagging accuracy. The only issue is handling the out-of-vocabulary (OOV) case. The tagger is trained on labeled training data. However, there are cases where lexical items in test data are not present in training data. Therefore, we have very little information to predict the tag of an OOV word. The straightforward solution for the OOV is based on frequency of tags. That is assigning the highest frequency tag for the OOV word. More advanced solution might use the suffix/prefix of word or the prior tag sequences. For example, the suffix *-ly* i.e. *beautifully, quickly, lovely* etc, indicates an adverb, while the suffix *-ive* i.e. *attractive, possessive* indicates an adjective. Also, the information about tag sequences indicates that after determiner is usually a noun.

1.5 Challenges in POS tagging

The first challenge for POS tagging is the training data. Currently, all supervised taggers outperform unsupervised ones. Supervised algorithms for POS tagger perform as accurate as 97% for English (Ratnaparkhi 1996), French (Denis and Sagot 2012) and many other languages. However, supervised learning needs manually annotated data which is time consuming and costly to construct. There are approximately 7000 languages in the world but very small fraction (around 30 languages) have sufficient POS manually annotated data for building reliable supervised POS tagger. Unsupervised POS tagging, on the other hand, does not need any manually annotated data. It is more or less equivalent with unsupervised clustering task. That is, they tried to group words into syntactic cluster. The hypothesis is that words in the same cluster must share the same POS. However, there is a huge gap between supervised and unsupervised learning accuracy.

Current POS taggers are language oriented; there is a lack of consensus about the tagset. Tag set are adapted to each language, *an obstacle for cross-language processing*. For example, when calculating syntactic similarity between two languages, we need to compare tag sequence similarity. However, since tag sets for each language are different, they are incomparable. Another example is when working with a multilingual environment such as the World Wide Web, giving a solution that can work for every language is in high demand. However, if we keep individual tagsets for each language, we might have to manually or semi-automatically map tagsets between languages pairwise.

1.6 Universal Tagger

In this thesis, we are going to investigate a solution to address the current challenges for POS tagging using a *Universal Tagger*. Supervised styles, as mentioned above, are impossible for our Universal Tagger, which aims at building taggers not only for resource-rich but also for resource-poor languages. Completely unsupervised style is possible but low accuracy makes unsupervised POS tagger impractical. In this paper, we would like to investigate an unsupervised approach but additionally, exploit parallel data to copy tag information from resource-rich to resource-poor languages. Parallel data is acquired from a parallel corpus which contains many pairs of source sentences and their translation to the target language. We can use parallel data to bridge between languages and transfer annotated data from one language to the other. The intuition is that, for many resource-poor languages, there is no manually POS annotated data. Such data involves the intensive work of linguist, but parallel data is easier to acquire.

The development of parallel data motivates us to build the Universal Tagger. Multilingual government documents, film subtitles, and a large amount of translation memory from books and news-papers, which are the source for parallel data, are becoming more and more widely available. Not only the size but also the language coverage of parallel data has improved drastically. The era of English dominating one side of parallel data is shifting to a far wider range of languages.

Moreover, the Universal Tagger aims at pushing the boundary for cross-lingual natural language processing (NLP). Thus, the output of the Universal Tagger for each language must be comparable. To do this, we must employ the same tagset across languages. In this ways, we resolve the traditional "language oriented" issues of the conventional POS tagger.

1.7 Main contributions

In this thesis, we concentrate on the single task of unsupervised multilingual POS tagging (Universal Tagger) which exploits parallel data. We give thorough reviews at related works on both traditional approach (monolingual) and modern approach (multilingual). We show both strong and weak points of these approaches in Chapter 2.

On Chapter 3 we are going to build a Universal Tagger which employs the consensus 12 Universal Tagset (Petrov *et al.* 2012) which will enable cross-lingual comparison. Given a tagger for the source language and source-target language parallel data, we are able to build a tagger for the target language. We evaluate this on the same test data on the same 8 languages with the-state-of-the-art (Das and Petrov 2011) using the same per-token accuracy metric. On average, our system performs on par with theirs but is penalized by using less data. Moreover, our method is much

less sophisticated and runs much faster.

Out of 8 languages, we perform better at 4 languages which are in the same language family tree (Germanic) with source language (English). This fact motivates us for Chapter 4 which we dedicate for investigating the effectiveness of choosing different source languages. It turns out that English is rarely the best source language for many cases. We are also able to build a model that can predict the best source language just based on monolingual data. This model performs better than always fixing a single best source language. The further accurate prediction can be acquired if we additionally use multilingual data from the parallel corpus. Finally, we show that, if multiple source languages are available we can even get further improvement by incorporating information from multiple source languages.

Chapter 5 contains the conclusion and a discussion of future work. We organize future work into different categories chronologically. That is: (1) immediate work which might take few weeks to finish; (2) near future work which might take a few months, and (3) longer future work which might take years.

Chapter 2

Background and Literature Review

This chapter reviews methods for monolingual and multilingual POS tagging. It provides background knowledge that is necessary for the next chapters. Monolingual POS tagging aims at building a tagger for a single language, and it does not take into account the relationship with other languages. Monolingual POS tagging is language-specific, which might used languages rules/heuristics to improve performance. Multilingual POS tagging, on the other hand, favoring a language-independent approach. It takes into account the relation between languages and aims at building taggers for many languages. More specifically, multilingual taggers build a tagger for one language based on data from other languages. Monolingual POS taggers exploit supervised, semi-supervised or unsupervised methods depending on the availability of data. Multilingual POS tagger, in contrast, is more focused on unsupervised methods with extra information from other (resource-rich) languages.

2.1 Monolingual POS Tagger

As mentioned before, POS tagger has gained much attention in the past and has achieved great success. Many algorithms for POS tagging have been developed over time. In this section, we are going to review some of them. Moreover, all POS taggers can be divided into 3 categories: *supervised, unsupervised and semi-supervised*. Current best taggers for English are showed in Table 2.1[1]. All these taggers used the Penn Treebank Tagset (Marcus *et al.* 1993) and are evaluated on the Wall Street Journal (WSJ) data set. The performance is measured based on per-token accuracy. Moreover, the accuracy for all tokens and unknown tokens are also given. As expected, all the best systems use some of the label data as training data in supervised or semi-supervised style. Moreover, some systems also used external resources to handle OOV or improve the accuracy.

[1]http://aclweb.org/aclwiki/index.php?title=POS_Tagging_(State_of_the_art)

System name	Short description	All tokens	OOV
TnT	Hidden markov model	96.46%	85.86%
MElt	MEMM with external lexical information	96.96%	91.29%
GENiA Tagger**	Maximum entropy cyclic dependency network	97.05%	NA
Maxent easiest-first	Maximum entropy bidirectional easiest-first inference	97.15%	NA
SVMTool	SVM-based tagger and tagger generator	97.16%	89.01%
Stanford Tagger 1.0	Maximum entropy cyclic dependency network	97.24%	89.04%
Stanford Tagger 2.0	Maximum entropy cyclic dependency network	97.29%	89.70%
Stanford Tagger 2.0	Maximum entropy cyclic dependency network	97.32%	90.79%
SCCN	Semi-supervised condensed nearest neighbor	97.50%	NA

Table 2.1: List of best English taggers (all > 96% on WSJ test set)

2.1.1 Supervised

In this part we review some supervised tagging algorithms. Each algorithm is in a different style, that is, *rule based, probabilistic, feature based* and some *other approaches*.

Transformation Based Tagging - a rule based approach

Transformation base tagging (TBT) (Brill 1995) uses rules to tag. For example, a rule could be, "A word after determiner is Noun" or "a/an/the are determiner". These rules are acquired from training data. So, basically, TBT contains two components, the **learner** and the **tagger**. The learner learns transformation rules from training data. The tagger uses these rules to tag.

The learner Steps for the learner are visualized in Figure 2.1. More detail is as follows:

- Strip the tag off the annotated data but keep the original data to evaluate.
- Initialize label for stripped data by some simple method such as, using frequency or any available tagger.

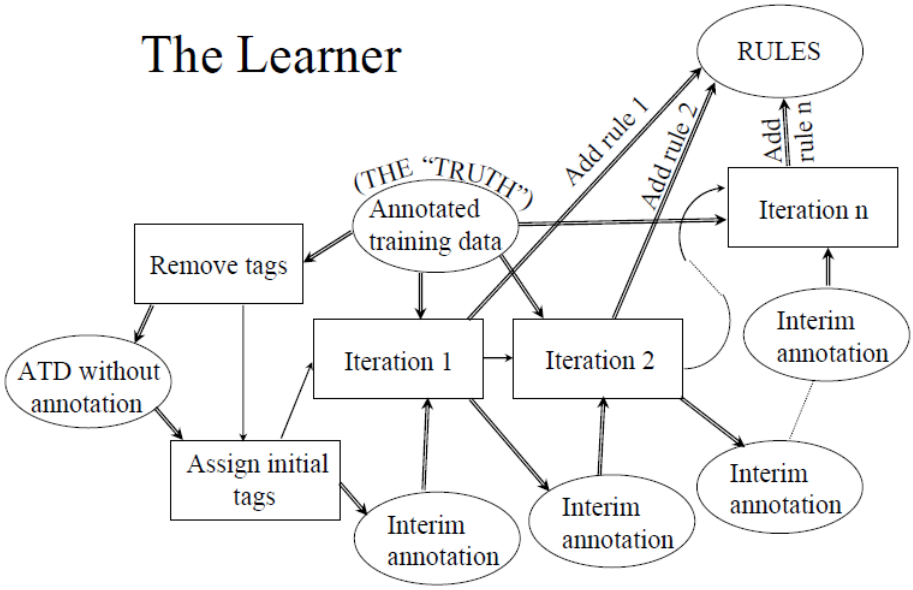

Figure 2.1: Learner component of TBT

- Start with an empty set of selected rule S.
- Repeat until the stopping criterion is applied. Each iteration, the input are the truth data, pool of possible rules and intermediate data from previous iteration. For each rule r in pool of rules, compute its contribution as follow:

  $$contrib(r) = c_{improved}(r) - c_{worsened}(r)$$

  where $c_{improved}(r)$ is the number of correct items which originally incorrectly tagged and $c_{worsened}(r)$ is the number of incorrect items (originally correctly tagged) after applying rule r. Afterward, they select a rule r which has the biggest contribution $contrib(r)$ and add to the final set of selected rules S. The pool of rules is acquired automatically or manually. Rules are designed in the form, "*change the tag from A to B / condition*". For example, the rule "*NN → NNS / preceded by NN VBP*", say: change the tag at current position from *NN* to *NNS* if two previous tags are *NN* and *VBP*. This step stop when no improvement can be made or the improvement less than a threshold.
- Output set of rule S

The Tagger The input for tagger is unlabeled data and set of rules S from the learner. Steps for the tagger are straightforward.

- Initialize label for lexical items in the same way as the learner did.

- Loop for all the rules. For each rule r, apply this rule to the whole intermediate to change some tag.
- The last intermediate data is the output.

TBT is one of the oldest and a very straightforward algorithm, however, it might be used to improve the accuracy of the original tagger. That is, use the original tagger to initialize the learner. Dien and Kiem (2003) initialize the learner by using an available English tagger. They also exploit external information from Vietnamese-English parallel data. The English tagger's performance is significantly improved, accuracy improved from 95.4% to 97.5% (absolute).

The Brill Tagger (Brill 1995) is one of the implementation of transformation base learning. TBT is straightforward but, the learning and tagging is quite slow. Performance of TBT largely depends on having pool of rule templates which are developed by linguist or acquired from machine learning algorithm on annotated corpus. Moreover, compared with other taggers, this approach has rather poorer performance.

Hidden Markov Model - a probabilistic approach

The idea of Hidden Markov Model (HMM) tagger is very similar to noisy channel or translation model in statistical machine translation. The difference is that the output is not a target text but a sequence of tags. Given a sentence which is a sequence of words $W = w_1w_2w_3....w_n$, we need to find the corresponding sequence of tags $T = t_1t_2t_3..t_n$ which maximizes the following probability.

$$Tags = \arg\max_{t_i} \ P(t_1t_2t_3..t_n|w_1w_2w_3....w_n) = \arg\max_{T} \ P(T|W)$$

According to Bayes' rules

$$P(T|W) = \frac{P(W|T) \times P(T)}{P(W)}$$

Since we're choosing tags, $P(W)$ is not considered, therefore

$$Tags = \arg\max_{T} \ P(T|W) = \arg\max_{T} \ P(W|T) \times P(T)$$

where

$$P(W|T) \times P(T) = P(w_1w_2..w_n|t_1t_2..t_n) \times P(t_1t_2..t_n)$$

According to chain rule

$$P(W|T)\times P(T) = P(w_1|t_1t_2..t_n)\times P(w_2|t_1t_2..t_n)\times P(w_3|t_1t_2..t_n)\times....\times P(w_n|t_1t_2..t_n)\times$$

$$P(t_n|t_1..t_{n-1}) \times P(t_{n-1}|t_1..t_{n-2}) \times ... \times P(t_1)$$

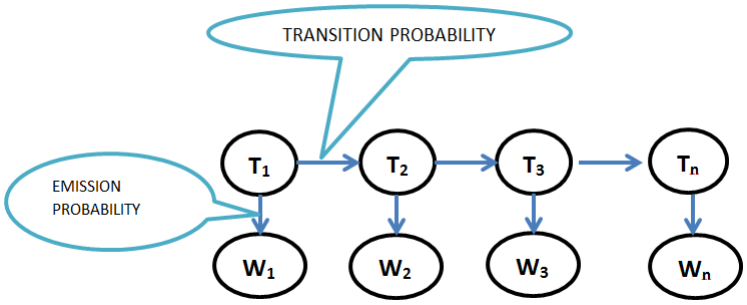

Figure 2.2: Emission and transition probabilities. W_i is a word and T_i is a tag

$$= \prod_{i=1}^{n} P(w_i|w_1t_1....w_{i-1}t_{i-1}) \times P(t_i|w_1t_1....w_{i-1}t_{i-1}) \tag{2.1}$$

With the assumption that the probability of word doesn't depend on the context but only on current tag, we have:

$$P(w_i|w_1t_1....w_{i-1}t_{i-1}) \approx P(w_i|t_i) \tag{2.2}$$

With the assumption that the choice of tag depends on limited history (for example bigram context), we have:

$$P(t_i|w_1t_1....w_{i-1}t_{i-1}) \approx P(t_i|t_{i-1}) \tag{2.3}$$

From equation (2.1), (2.2) and (2.3):

$$Tag\ Sequence = \arg\max_{T}\ P(W|T) \times P(T) \approx \arg\max_{t_i}\ \prod_{i=1}^{n} P(w_i|t_i) \times P(t_i|t_{i-1}) \tag{2.4}$$

$P(w_i|t_i)$ is called the *emission probability* and $P(t_i|t_{i-1})$ is called the *transition probability* as shown in Figure 2.2. We estimate these probabilities from training data and then smooth them using method such as interpolation (Reinsch 1967) or Good Turing (Gale 1994). Smoothing is needed because training data might not cover everything that appears in test data.

HMM tagging is not as trivial as transformation based tagging. The straightforward tagging solution is that we list all possible tag sequences and calculate the probability of each sequence based on equation 2.4. The result is the tag sequence that gave the highest probability. However, this solution is not realistic because of the exponential complexity. Therefore, people use dynamic programming algorithms such as Viterbi (Ryan and Nudd 1993) or beam search (Reddy 1976).

TNT (Brants 2000) is an implementation of HMM approach which uses trigram tag model, i.e. $P(t_i|t_{i-1}t_{i-2})$ instead of bigram tag model as mentioned above. Back to Table 2.1 (page 7) , we can see that TNT performs well and achieves comparable result with the state-of-the-art tagger.

Maximum Entropy - a feature based approach

We aim at maximizing the following probability

$$\arg\max_{t_i} \ P(t_1 t_2 t_3 .. t_n | w_1 w_2 w_3 .... w_n) = \arg\max_{T} \ P(T|W)$$

HMM tried to maximize the joint probability $P(W,T)$

$$\arg\max_{T} \ P(T|W) = \arg\max_{T} \ P(W|T) \times P(T) = \arg\max_{T} \ P(W,T)$$

Maximum Entropy Tagging (MaxEnt) (Ratnaparkhi 1996) directly targets maximizing conditional probability $P(T|W)$. The disadvantage of HMM tagging is that it cannot incorporate many source of evidence from the text. It just uses lexical and tag information which denoting in emission and transition probability. The MaxEnt model resolves this disadvantage. We can embed as much information as we want in to this MaxEnt model. These information might vary from position, lexical, tag, global document and so forth. Each of these pieces of information are call features. This is the reason why Maximum Entropy tagger belongs to the feature based approach. Formally, a feature is defined as:

$$f_i(C,t) = \begin{cases} 1 & \text{if } context\ C\ is\ satisfied \\ 0 & otherwise \end{cases}$$

The feature could be complicated or simple, but a feature always in the form, context C and tag t. For example a feature "*suffix = -ing and tag = VBG*" can be denoted as

$$f_i(C,t) = \begin{cases} 1 & \text{if } \ suffix = \ -ing \ \& \ t = VBG \\ 0 & otherwise \end{cases}$$

The model using these features has to obey the **constraints**, that is, expected value of feature f_i which is $E_p f_i$ must comply with training data.

$$E_p f_i = E_{p'} f_i$$

Where

$$E_{p'} f_i = \frac{1}{N} \sum_{j=1}^{N} f_i(C_j, t_j)$$

$E_{p'} f_i$ is the expected value of feature f_i which is derived from training data (C_1, t_1),(C_2, t_2) .. (C_n, t_n). This list can be derived easily from manually annotated data. Apply Lagrangian Multipliers on these constraints, we derived MaxEnt model which has the form as follows:

$$p(t|C) = \frac{1}{Z(C)} exp\Big(\sum_{i=1}^{n} \lambda_i f_i(C,t) \Big) \tag{2.5}$$

Where

- f_i is a feature function as defined above
- C is the context, t is tag
- $Z(C)$ is the normalization factor to ensure proper distribution probability.
- λ_i is the weight for features f_i

The more common name for this model is "log-linear model". We use Generalized Iterative Scaling (GIS) (Darroch and Ratcliff 1972) to estimate λ_i. The value of λ_i shows the importance of feature f_i. This value has to obey the *constraints* applied for feature f_i. The GIS algorithm is shown in Algorithm 1.

Algorithm 1 Finding λ_i

$\lambda_i^{(0)} \leftarrow 0$
while *not convergence* **do**
  $C \leftarrow max_{C,t} \sum_{i=1}^{n} f_i(C,t)$
  $\lambda_i^{(t+1)} \leftarrow \lambda_i^t + \frac{1}{C} log \frac{E_{p'} f_i}{E_{p^{(t)}} f_i}$
end while

It is very important to notice that the constraints cannot uniquely define the model in Equation (2.5). So, from all models that satisfy constraints, we choose a model that *maximizes entropy*, that is, as uniform as possible. Moreover, tag probability truly depends on the context which is described by features, therefore:

$$Tag\ Sequence = \arg\max_{T}\ P(T|W) = \arg\max_{T}; P(T|C) \approx \arg\max_{t_i}; \prod_{i=1}^{n} p(t_i|C_i) \quad (2.6)$$

Finally, the most probable sequence of tags (*result tag sequence*) is obtained by performing beam search (Reddy 1976) or Viterbi (Ryan and Nudd 1993) algorithms to maximize equation (2.6).

The most common implementation of MaxEnt tagger uses features from two words back and ahead, the current word, and two tags back. That is, context C $= (w_{i-2}, t_{i-2}, w_{i-1}, t_{i-1}, w_i, w_{i+1}, w_{i+2})$. Features can be anything from this context. Therefore, the number of features can be numerous and degrade the performance. Effective implementation always uses some heuristic methods to limit the number of features. Stanford Maximum Entropy tagger (Toutanova *et al.* 2003a) is one of notable such implementation. From Table 2.1 (page 7), we can see that aside from Stanford MaxEnt, there are many other systems i.e. GENiA Tagger (Toutanova *et al.* 2003b) or Maxent easiest-first (Tsuruoka and Tsujii 2005) exploit MaxEnt idea and acquire very good result. This is somehow understandable because MaxEnt has the power to incorporate all features. In that way, we capture all evidence that could help to predict tags.

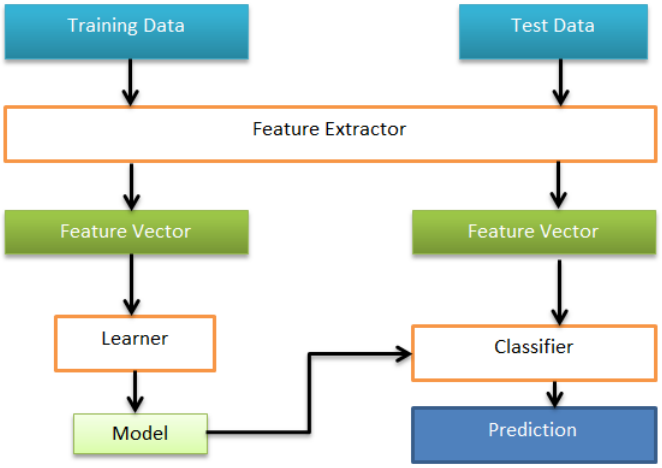

Figure 2.3: Steps to build a classifier

Other algorithms

Apart from the above mentioned approaches (rule based, probabilistic, feature based), there are other approaches for supervised POS taggers.

The *classifier base approach* is one of those. They build a classifier based on machine learning algorithms such as Naive Bayes, Decision Tree, kNN or more advanced methods: Support Vector Machines (SVM) or Conditional Random Fields (CRF) and so forth. Each lexical item (word) is converted into features, then a classifier is trained on this data. The test data is also converted into features. We run the classifier on test data and produce tag labels. Steps for classifier approach are described in Figure 2.3. Back to Table 2.1 (page 7), SVMtool (Gimenez and Marquez 2004) implement SVM machine learning algorithm performs very well on Wall Street Journal (WSJ) data set.

Parsing approach exploits the idea of building a constituency parse tree as in Figure 2.4. Afterward, we need to track down the rules to determine tags. For example, from rule NP $\rightarrow$ DT NN, we extract "*a*" (DT) and "*fly*" (NN). However, parsing task is more difficult than tagging and actually uses information from tagging processes. Nevertheless, parsing could be very helpful for tagging especially in the case where disambiguation needs a long range syntactic information as in Chinese or Japanese. Traditionally, a constituency parser is built from a tree bank using algorithms such as shift-reduce parsing (Yeh 1983). Modern approach does POS tagging and parsing at the same time in an incremental way (Hatori *et al.* 2012).

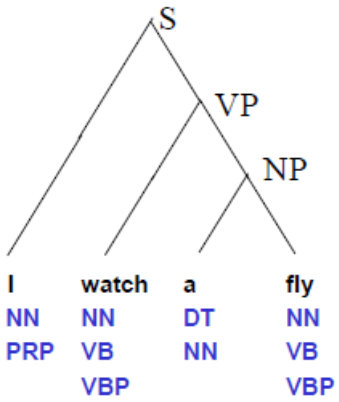

Figure 2.4: Constituency Parse Tree

2.1.2 Unsupervised

Unsupervised tagging does not use any labeled data for training. There are several well-known methods for unsupervised tagging.

Clustering

Words having the same syntactic properties are grouped into the same group (cluster). It is believed that words in the same cluster are likely to have the same POS tag. The first issue a clustering tagger has to tackle is *measuring syntactic similarity*. That is if each word is a vertex in a graph, and the weight of edges between vertexes is the syntactic similarity, then words which have the same POS label must be close to each other. Normally, similarity measurement algorithm is based on nearby content words. For example, in two sentences "*He bought a pair of shoes*" and "*He ate a banana*". Two words "*bought*" and "*ate*" are surrounded by "*He*" and "*a*", therefore "*buy*" and "*eat*" are likely to have the same POS tag. Schütze (1995) proposes to use distributional similarity for measurement. This approach is based on the hypothesis that, "we can understand a word by looking at its neighbor feature words". Afterward, a word is displayed as vectors of feature words. To compensate for the frequency effects, the weight for each feature word is calculated using a method such as tf.idf (Robertson 2004). Schütze (1995) considers the top n most frequent words as feature words. A word is displayed as:

$$\overrightarrow{wordA} = (w_1, w_2, w_3.....w_n)$$

where w_i is the weight of the i^{th} feature word and $w_i = 0$ if that feature word does not appear in the context of current $wordA$. The similarity between two words is the cosine similarity between the corresponding vectors.

Another problem with clustering tagger is **determining the number of clusters**. Defining that number beforehand might not be a good solution. We might force the algorithm to separate coherent cluster or to join unrelated one. On the other hand, letting the algorithm choose when to stop could result in a too specific or too general cluster.

Evaluation is also another major consideration. Normally, clustering algorithms are evaluated based on the perplexity (or entropy) of the cluster. In the case of tagging, we are expecting that all words in the same cluster have the same tag. Therefore, the lower the perplexity, the better. However, is it what we are looking for? The answer is no, we want to compare with gold standard test data to know the tagging performance. Johnson (2007) suggested a *many-to-one* evaluation. The induced tag for each cluster is the most frequent tag of the items in the cluster, consulting the gold standard data. However, there are cases where two clusters have the same tag. To resolve this issue, *one-to-one* evaluation puts the restriction that each gold tag corresponds to only one cluster. Normally, it is done by greedy matching, which aims at maximizing accuracy. Nevertheless, the number of clusters and gold tags are likely to be different. In that case, some clusters or gold tags will not be matched. However, both *many-to-one* and *one-to-one* evaluation schemes require gold data to find the most appropriate tag for each cluster. This is a chicken-and-egg problem since if we have gold data then we don't need to take an unsupervised approach. Besides, we can also use some heuristic method to determine the tag for each cluster such as cluster size (eg, the biggest cluster is Noun).

UnsuPOS[2] (Biemann 2006b) is an implementation of a clustering tagger. He divides words into two partitions: high and medium frequency words, medium and low frequency words. In each partition, he builds a different graph using a different similarity measurement. For high and medium frequency words, he uses distributional similarity as mentioned above. For the other partition, he calculates similarity between two words based on how many *direct* neighbors they have in common. The edge is made between words (vertex) if the weight is higher than a threshold. Afterward, the Chinese Whispers (Biemann 2006a) clustering algorithm is performed on both graphs. Steps for Chinese Whispers are simply as follows: (1) each node is assigned to a cluster; (2) sort the nodes in an random order; (3) join node to it most popular neighborhood where popularity is defined as the total weight of inside nodes; (4) repeat (3) until the number of cluster reaches the fix point.

Thereafter, on each partition, there is a set of clusters. It is worth noticing that there are many words in common (medium frequency words) between clusters. Then, Biemann (2006b) merges these two sets into a single set. Again, each cluster is considered as a vertex and the number of common words is the weight between vertexes. Chinese Whisper is applied again to acquire final clusters. Every word inside a cluster has cluster ID as the tag label. Finally, he build a tagging model

[2]http://wortschatz.uni-leipzig.de/~cbiemann/software/unsupos.html

using a trigram HMM.

Unsupervised tagging using a clustering algorithm is useful since it provides the first overview about syntactic categories of lexical items. In reality, it is usually used with other applications, for example, Søgaard (2011) uses unsupervised tagging information as the feature to build another tagger.

HMM unsupervised tagging

In a supervised HMM tagger, we estimate emission probability $p(w_i|t_i)$ and transition probability $p(t_i|history)$ from the labeled training data. In the unsupervised style, these two probabilities are estimated using the Baum-Welch (Baum *et al.* 1970) algorithm, which is a variant of the *Expectation Maximization (EM)* algorithm. It means that Baun-Welch can only output the local best but not the global optimal solution. Baum-Welch makes use of forward and backward algorithms (Zweig 1996). It uses the sequence of observations and outputs the most likely parameters (emission and transition probability) for the HMM. Roughly speaking, the steps for Baum-Welch are described in Algorithm 2.

Algorithm 2 Baum-Welch algorithm

```
Initialize emission probability P_Y
Initialize transition probabilityP_S
while not convergence do
  Compute forward probability
  (1) Using current setting of P_S and P_Y
  (2) Follow the procedure of trellis algorithms
  Compute backward probability
  (1) Same as forward probability
  (2) Start from the end
  for each emission/transition pair do
    Collect count base on forward & backward prob
  end for
  Re-estimate P_S and P_Y
end while
```

Feature based HMM (FB-HMM) (Berg-Kirkpatrick *et al.* 2010) is another variance of unsupervised HMM tagging. The emission probability model $P(w_i|t_i)$ is estimated using a log-linear model:

$$P(w_i|t_i) = \frac{exp(weight \times f(w_i, t_i)}{\sum_{x' \in Voc} exp(weight \times f(x, t_i))}$$

where Voc is the entire vocabulary and $f(w_i, t_i)$ is the feature as in supervised feature based learning. Again, this model can incorporate many source of evidence from the

	Model	da	nl	de	el	it	pt	es	sv	**Avg**
Without dic.	EM-HMM	68.7	57	75.9	65.8	63.7	62.9	71.5	68.4	66.7
	FB-HMM	69.1	65.1	81.3	71.8	68.1	78.4	80.2	70.1	73.0
With dic.	FB-HMM	93.1	94.7	93.5	96.6	96.4	94	95.8	85.5	93.7

Table 2.2: Comparison between with and without dictionary unsupervised POS tagger for 8 languages (Danish(da), Dutch(nl), German(de), Greek(el), Italian(it), Portuguese(pt), Spanish(es), Swedish(sv)

observation w_i. In the original paper, Berg-Kirkpatrick *et al.* (2010) use features including: (1) original word w_i; (2) whether w_i contains digit or not; (3) whether w_i contains a hyphen or not; (4) whether the first character is capital or not and (5) the features from the previous 3 words. The feature based HMM performed better than EM-HMM, simply because it exploits more information. Table 2.2 show the margins of 6.3 % (absolute) in average accuracy between FB-HMM (73%) and EM-HMM (66.7%).

In the unsupervised HMM approach, we assume that we know the tag set and *possible tags for each word*. We can get this list of possible tags from the dictionary or from training data. If we don't have this resource, the system would be completely unsupervised. However, without this list, the performance would degrade substantially. Das and Petrov (2011) use the same tagset for different languages, the comparison of supervised, completely unsupervised (without dictionary EM-HMM and FB-HMM), and unsupervised with dictionary (with dictionary FB-HMM) tagger are given in Table 2.2. The effectiveness of the gold dictionary is clearly visible. The average accuracy increases from 73.0% to 93.7% (absolute).

2.1.3 Semi-supervised

Semi-supervised learning is the combination of supervised and unsupervised method. The amount of labeled data is not enough to build a reliable supervised system. On the other hand, unlabeled data is widely available and easy to access in large quantity. In this circumstance, semi-supervised learning is the solution. Moreover, even in the case where labeled data is big enough, combining with unlabeled data might give better performance.

Self-training, Tri-training approach

The most straightforward approach to semi-supervised tagging is self-training. That is, we train a supervised classifier c from labeled data L. This model is used to label unlabeled data U, so U will become labeled data L'. Afterward, we merge L' and the gold labeled data L, and train new classifier c' on them. The process is repeated until no new classifier is constructed. Tri-training (Zhou and Li 2005) is the extension

of self-training. Instead of using just one classifier, tri-training uses three classifiers. All three classifiers are originally trained on *bootstrapped* gold data. Bootstrapping is a technique for re-sampling the data. It is very important to use bootstrapped data, because if training data for three classifiers is not diverse, the three classifiers would be the same and tri-training would become self-training. When labeling a data item from unlabeled data U, two classifiers would help to determine the label. The intuition here is that when two classifiers agree upon a label, it is likely to be correct. The algorithms stop when no new classifiers are built. The detailed steps are described in Algorithm 3.

Algorithm 3 Tri-training algorithm

1: $L_i \leftarrow bootstrap(L)$: i = 1..3
2: $c_i \leftarrow train\_classifier(L_i)$: i = 1..3
3: **repeat**
4: **for** i : 1..3 **do**
5: **for** $x \in Unlabeled\ data$ **do**
6: **if** $c_j(x) = c_k(x)$ & $(j, k : 1..3)$ & $(j, k \neq i)$ **then**
7: $L_i \leftarrow L_i \cup (x, c_j(x))$
8: **end if**
9: **end for**
10: $c_i \leftarrow train\_classifier(L_i)$: i = 1..3
11: **end for**
12: **until** c_i not change
13: $Final\ classifier \leftarrow vote\ of\ c_i$: i= 1..3

Søgaard (2010) suggested a simple improvement for tri-training, that is, tri-training with disagreement. He just simply replaced the condition $if\ c_j(x) = c_k(x)$ (in line 6, Algorithm 3) with condition, $if\ c_j(x) = c_k(x) \neq c_i(x)$. The accuracy and especially **speed** improved. It might be because the new condition just focuses on the weak point of the classifier. The advantage of self-training is the method's simplicity. However, in practice, without exploiting extra information, self-training gives very little improvement. Table 2.3 shows the result of self-training, tri-training and tri-training with disagreement on the same WSJ data set based on the two tagger SVM Tool (Gimenez and Marquez 2004) and MaxEnt (Toutanova *et al.* 2003a). Tri-training with disagreement performs exactly the same as Tri-training on both models. However, due to the modification in the condition, Tri-training with disagreement runs faster. The improvement of Tri-training over self-training is not clearly visible.

Semi-supervised condensed nearest neighbor

Looking back at Table 2.1 (page 7), the semi-supervised condensed nearest neighbor (SCCN) tagger (Søgaard 2011) gave the highest accuracy (97.5%) on the WSJ

	Original	Self-training	Tri-training	Tri-training with dis.
SVM Tool	97.15	97.26	97.27	97.27
MaxEnt	96.31	96.36	96.36	96.36

Table 2.3: Comparing self-training, tri-training and tri-training with disagreement

Gold Label	Supervised label	Unsupervised label
NN	NN	14
VB	VBZ	121
DET	DET	11
...	...	...

Table 2.4: Sample data point in train data

data set. Søgaard (2011) exploited the vast amount of unlabeled data to find a better center point for each cluster. SCCN is based on the kNN machine learning algorithm. That is, the label is chosen based on the vote of k nearest neighbors. Similarity measurement between data points are first to be put into consideration. Each data point is translated into a vector $\overrightarrow{x} = (m, n)$, where m is the label given by the supervised tagger SVMTool (Gimenez and Marquez 2004) and n is the label/clusterID given by unsupervised clustering tagger UnSuPos (Biemann 2006b). Examples of data points for training are given in Table 2.4. The similarity between two data points is calculated based on euclidean distance between two corresponding vectors.

kNN is considered as a lazy method as it does not actually build a model. Every time, it needs to scan through all the data points again. Therefore, the complexity of kNN for each tag is proportional to the number of labeled data points. In this way, it is not possible for kNN to run on big data sets. The condensed nearest neighbor (CNN) (Søgaard 2011) is an intuitive solution. CNN tries to condense the data by filtering out data points which are not a good representation. The condense technique is straightforward, as shown in Algorithm 4. L is the labeled data points, where x_i is data and y_i is the tag. The initial $classifier$ is trained on L. Søgaard (2011) uses this classifier to discard all the data points that are correctly tagged. An extension of CNN is weakened condensed nearest neighbor (WCNN) where they just discard any high confidence correctly classified items. It means that line 4 of Algorithm 4 will be replaced by **if** $Classifier(x_i) \neq y_i$ *or* $Prob(x_i, y_i) < threshold$. where $Prob(x_i, y_i)$ denotes the classifier confidence. Basically, CNN or WCNN tries to find the best representative points for each cluster. Ideally, each cluster will find one and just one single point located at the center of cluster. However, since the amount of labeled training data is not big enough, it is unlikely to cover this point. This is exactly where **unlabeled data comes into play**, with the advantages of gigantic size, it is high chance that unlabeled data points would be close enough to the center point.

Algorithm 4 Condense Nearest Neighbor algorithm

1: $L = \{(x_1, y_1), (x_2, y_2)....(x_n, y_n)\}$
2: $C \leftarrow \{\}$
3: **for** $(x_i, y_i) \in L$ **do**
4: **if** $Classifier(x_i) \neq y_i$ **then**
5: $C \leftarrow C \cup (x_i, y_i)$
6: **end if**
7: **end for**

Semi-supervised weakened condensed nearest neighbor is described in Algorithm 5. Firstly, they obtain condensed data points C using WCCN. For each unlabeled data item, they try to tag it using C. If it is tagged with high confidence (greater than some threshold, i.e. 0.9), they add this point and its label to C. In the last step, they apply the condense technique again on C to get final condensed set of data points C'. Using C' to tag is as trivial as in any kNN algorithms. For the WSJ dataset, this method actually gave the highest accuracym 97.5%. Moreover, it also runs extremely quickly thanks to very small number of data point in C'. Søgaard (2011) pointed out that it condenses the 1.2 million data points of the Brown Corpus to just 2249 items.

Algorithm 5 Semi-Supervised Weakened Condense Nearest Neighbor algorithm

1: $L = \{(x_1, y_1), (x_2, y_2)....(x_n, y_n)\}$
2: $U = \{(x'_1), (x'_2), ...(x'_m)\}$
3: $C \leftarrow WCNN(L)$
4: **for** $(x'_i) \in U$ **do**
5: **if** $TagProb(x'_i) > threshold$ **then**
6: $C \leftarrow C \cup (x'_i, y'_i)$
7: **end if**
8: **end for**
9: $C' \leftarrow WCNN(C)$

2.2 Multilingual POS Tagger

This section reviews methods for creating taggers for multiple languages based on parallel/comparable data. Comparable documents contain the *rough* translation of each other, for example, Wikipedia documents for different languages are comparable. Yarowsky and Ngai (2001), Xi and Hwa (2005), Snyder *et al.* (2008) and Das and Petrov (2011) exploit this idea. The following is the overview of their approaches.

2.2.1 Typologically related languages approach

This approach restricts to very closely related language such as Czech and Russian, Telugu and Kannada and so forth. Hana *et al.* (2004), Feldman *et al.* (2006), Reddy and Sharoff (2011) have exploited this idea. HMM taggers need emission probabilities and transition probabilities. The intuition here is that if two languages are typologically related, **their transition probabilities would be very similar**, that is, the tag sequences between these languages are similar to each other. Therefore, we can use transition probability interchangeably. Reddy and Sharoff (2011) built a tagger for Kannada from Telugu. Kannada and Telugu are both spoken in India by 35 million and 70 million people respectively. Telugu are more widely spoken and has better resources. Steps to construct Kannada tagger is as follows:

1. Collect large corpora for Kannada and Telugu
2. Tag the Telugu side using the available tagger
3. Calculate the Telugu transition probabilities $p(t_i|t_{i-1}, t_{i-2})$
4. Estimate Kannada emission probabilities $p(w_i|t_i)$
5. Build Kannada tagger from steps 3 and 4

It is worth noticing that this method does not need to collect parallel data of Kannada and Telugu. Monolingual or comparable corpora would work in this case. Hence, Reddy and Sharoff (2011) used Corpus Factory (Kilgarriff *et al.* 2010) to crawl monolingual data from the internet. Telugu has POS manually annotated data. Therefore, constructing a tagger, tagging and calculating transition probability (Step 2 and 3) is not a problem. However, step 4 is not trivial. The first solution is that, we can also estimate Kannada emission probability $p_K(w_i|t_i)$ via Telugu emission probability $p_T(w'_i|t_i)$ where w'_i and w_i are very similar to each other. Kannada and Telugu are very closely related languages, therefore, they share many lexical item in common. It is possible to find the closest word w'_i of w_i using any approximate string matching (fuzzy matching). Reddy and Sharoff (2011) also suggested using uniform distribution of all possible tags for emission probability. For each words, they get the list of possible tags from dictionary, and then assume that the tags are distributed uniformly. This turn out to be quite effective. Table 2.5 shows some results. The emission probability of model 1 is based on string approximation. It gave an encouraging result since they actually did not use any resource aside from monolingual data. Model 2 is surprisingly high. However, model 2 use gold dictionary to look for possible tags of each words which is hard to acquire. Model 3 uses actual emission probabilities from the target language. That is, we estimate emission probabilities using a supervised target language (Kannada) tagger. Model 3 is added for comparison purposes only. Model 4 is more or less the upper bound that the other models can get. Transition and emission probabilities are estimated directly from the Kannada side (using supervised tagger). Remarkably, model 3 and 4 gave very close result. This

Model	Transition Prob.	Emission Prob.	F-measure
1	From source language	Approximate string matching	56.88
2	From source language	Uniform distribution target tags	75.10
3	From source language	Target language emission Prob.	77.63
4	From target language	Target language emission Prob.	77.66

Table 2.5: Various tagging model exploit topologically related language

consolidates the hypothesis that "for closely related languages, transition probability are interchangeable".

2.2.2 Unsupervised simultaneously tagging approach

A previous approach used typologically related languages. It assumed the presence of tagger or labeled data of a source language and then copied transition probabilities to the target language. This approach, however, does not need any labeled data in both languages, words are tagged **simultaneously** (Snyder *et al.* 2008). Firstly, alignment are run to align words for each sentences. Word alignment exploits IBM models (1..5) and will be trained on large amount of parallel data. The output of the aligner is the list of mappings between words from source and target sentence going with confidence. We discard low confidence, cross-edges and multiple mapping. So, the remaining is high confidence one-to-one word alignment. These word pairs would be the observation of the model. Thus, for each sentence pair, just some words are aligned and the rest (the major) is unaligned. They try to maximize the following probability given the observed alignment and words.

$$\begin{aligned} P(x_1, x_2...x_m, y_1, y_2..y_n|a, \phi, \phi', w) = & \prod_{unaligned\ i} \phi_{x_{i-1}}(x_i) \\ \times \prod_{unaligned\ j} \phi'_{y_{i-1}}(y_j) \times & \prod_{(i,j)\in a} P(x_i, y_j|x_{i-1}, y_{j-1}, \phi, \phi', w) \end{aligned} \tag{2.7}$$

where $x_1, x_2...x_m$ is a tag sequence of the source language, $y_1, y_2..y_n$ is a tag sequence of the target language, a is the alignment, ϕ and ϕ' are transition probabilities of source and target language respectively; and w is the bilingual coupling distribution $w : T \times T'$ where T and T' are tag sets of the two languages. They infer equation (2.7) using Gibbs Sampling. They sample (1) the tag sequence (x_i, y_j), (2) monolingual transition probability ϕ and ϕ', (3) coupling distribution w and then integrate over emission probabilities. The model is evaluated on 4 languages: English, Bulgarian, Slovene and Serbian. The baseline for each language is calculated using a monolingual unsupervised HMM with gold dictionary. That is, possible tags for each word are provided. Table 2.6 shows the constant improvement of bilingual simultaneous tagging over monolingual tagging. Of particular note, tagging accuracy of Slovene

	Monolingual	**Simultaneously**	**Improvement**
EN	90.71	91.01	+0.3
SR	85.05	90.06	+5.03
EN	90.71	92	+1.29
BG	88.88	94.48	+5.61
EN	90.71	92.01	+1.3
SL	87.41	88.54	+1.13
SL	87.41	95.1	+7.69
SR	85.05	91.75	+6.7
BG	88.88	91.95	+3.08
SR	85.05	86.58	+1.53
BG	88.88	90.91	+2.04
SL	87.41	88.2	+0.79

Table 2.6: Per-token tagger accuracy with gold dictionary for unsupervised monolingual and bilingual simultaneously tagging

improves 7.69% when paired with Serbian which is very closely related language, but only 1.3% when going with English. Thus, we can see that choosing the right language pair would significantly improve the performance.

They further reduce the size of the gold dictionary to just the 100 most frequent words. The effectiveness of bilingual tagging is highlighted. Table 2.7 shows the huge improvement of bilingual simultaneously tagging. This table also highlights the differences in performance when changing the source language. For example, pairing BG with SR improves SR only 1.53% when full size dictionary is used (Table 2.6), however, improve $\approx 16\%$ on reduced dictionary (Table 2.7).

2.2.3 Tag projection approach

A number of studies have used *tag projection* to copy tag information from a resource-rich to resource-poor language, based on word alignments in a parallel corpus. After alignment, the resource-rich language is tagged, and tags are projected from the source language to the target language based on alignment (Yarowsky and Ngai 2001) and (Das and Petrov 2011). Das and Petrov (2011) achieved the current state-of-the-art for unsupervised tagging by exploiting high confidence (> 0.9) alignments to copy tags from the source language to the target language.

Graph-based label propagation was used to automatically produce more labelled training data. First, a graph was constructed in which each vertex corresponds to a unique trigram in the target language, and edge weights represent the syntactic (POS) similarity between two vertices. The syntactic similarity function is calculated from the window of two words around the target words. The features for trigram $x_2x_3x_4$

	Monolingual	**Simultaneously**	**Improvement**
EN	63.57	68.22	+4.66
SR	41.14	54.73	**+13.59**
EN	63.57	71.34	+7.78
BG	53.19	62.55	**+9.37**
EN	63.57	66.48	+2.91
SL	49.9	53.77	+3.88
SL	49.9	59.68	**+9.78**
SR	41.14	54.08	**+12.94**
BG	53.19	54.22	+1.04
SR	41.14	56.91	**+15.77**
BG	53.19	55.88	+2.7
SL	49.9	58.5	+8.6

Table 2.7: Per-token accuracy with reduced gold dictionary size for unsupervised monolingual and bilingual simultaneously tagging. Huge improvements are in bold

Description	Features
Trigram + Context	$x_1x_2x_3x_4x_5$
Trigram	$x_2x_3x_4$
Left Context	x_1x_2
Right Context	x_4x_5
Center Word	x_3
Trigram - Center Word	x_2x_4
Left Word + Right Context	$x_2x_4x_5$
Left Context + Right Word	$x_1x_2x_4$
Suffix	hasSuffix(x_3)

Table 2.8: Features used for computing similarity between trigram vertex

in the sequence of $x_1x_2x_3x_4x_5$ are described in Table 2.8[3]. The idea here is that they expect the same part-of-speech for words used in the same context. For example, two trigrams "he-has-lunch" and "he-gets-lunch" share the same $x_2 = he, x_4 = lunch$ therefore, "has" and "gets" are likely to have the same POS tag.

Labels were then propagated by optimizing a convex function to favor the same tags for closely related nodes while keeping a uniform tag distribution for unrelated nodes. They extract tag dictionary from the automatically labelled data. This dictionary, in turn, was used to constrain a feature-based HMM tagger (Berg-Kirkpatrick *et al.* 2010).

Das and Petrov (2011) experimented on 8 languages: Danish(da), Dutch(nl), German(de), Greek(el), Italian(it), Portuguese(pt), Spanish(es), Swedish(sv). They used parallel data from Europarl (Koehn 2005) and the ODS United Nations dataset which has English in the source side. They evaluated target language tagger performance on gold test data mainly from the CoNLL-X and CoNLL 2007 shared tasks. Table 3.6 (page 38) shows the detailed source and size for each gold test data set. To avoid mapping between different tagsets, they use consensus 12 Universal Tags (Petrov *et al.* 2012) across languages.

Das and Petrov (2011) experimented with 7 different models.

1. Unsupervised EM-HMM, which is a traditional expectation maximization unsupervised HMM. EM-HMM is mentioned in section 2.1.2 (page 16).

2. Unsupervised Feature based - HMM is another style of unsupervised HMM which is also mentioned in section 2.1.2 (page 16)

3. Direct projection, that is, tags are projected from source language to target language. Unaligned words are tagged with the most frequent tag. Afterward, supervised POS tagging is trained on this data.

4. No label propagation. The dictionary is extracted from direct projected tags. The graph-base label propagation step is not performed.

5. With label propagation. The full model, dictionary is extracted after label propagation step.

6. With gold dictionary. Gold dictionary is extracted from annotated data. This dictionary is fed into a feature base-HMM tagger as the constraint.

7. Supervised, the supervised POS tagger is trained on annotated data.

Model (1), (2), (3) are used as the baseline to compare. Comparing (4) and (5) show the effectiveness of label propagation. Model (6) show the upper bound of this

[3]hasSuffix(x_3) feature check the target word x_3 has suffix or not

approach. Model (7) is for comparing with supervised method. The performance of each model on 8 languages is described in Table 2.9.

As expected, unsupervised feature based HMM (model 2) performs better than the traditional EM-HMM (model 1). The tagger built on direct projection (model 3) is actually a weakly supervised [4] system, because the target language labels are not 100% correct, coming from the heuristic (projection) step. However, it still performs better than the completely unsupervised method, EM-HMM (model 1) and Feature-HMM (model 2). Label propagation (model 5) shows constant improvements over all 8 languages, improved 2% (absolute) compared with model 4 where label propagation was not employed. This improvement is mainly because performing label propagation enlarges the dictionary size and thus, improves the vocabulary coverage. This full model with Label Propagation (model 5) outperforms unsupervised methods (model 1, 2) and does better than direct projection (model 3) at 7 languages out of 8. It also narrows the gap between unsupervised and supervised methods (model 7). In this thesis, we will mainly compare with this work as it is considered as the state-of-the-art system for unsupervised multilingual POS tagger

Model	da	nl	de	el	it	pt	es	sv	Avg
(1) EM-HMM	68.7	57.0	75.9	65.8	63.7	62.9	71.5	68.4	66.7
(2) Feature-HMM	69.1	65.1	81.3	71.8	68.1	78.4	80.2	70.1	73.0
(3) Projection	73.6	77.0	**83.2**	79.3	79.7	82.6	80.1	74.7	78.8
(4) No LP	79.0	78.8	82.4	76.3	84.8	87.0	82.8	79.4	81.3
(5) With LP	**83.2**	**79.5**	82.8	**82.5**	**86.8**	**87.9**	**84.2**	**80.5**	**83.4**
(6) With gold Dic	93.1	94.7	93.5	96.6	96.4	94.0	95.8	85.5	93.7
(7) Supervised	96.9	94.9	98.2	97.8	95.8	97.2	96.8	94.8	96.6

Table 2.9: Accuracy of multiple models on 8 languages. The best unsupervised performance for each language is in bold

[4]Unlike supervised, weak supervised (or some time called distant supervised) learning doesn't have the actual gold training data, these data are induced from heuristic step and not completely correct.

Chapter 3

Universal Tagger

3.1 Idea and Motivation

We have reviewed a possible approach for monolingual and multilingual POS tagger in the previous chapter. In this chapter, we aim to create our own universal tagger for many languages. Currently, there are separate taggers for many languages such as English, French, German, Italian, Arabic and so forth, where manually annotated data are available. Petrov *et al.* (2012) show that the average accuracy for supervised learning for languages in table 3.1 is as high as 95,2% using the basic TNT tagger (Brants 2000). However, not many languages have enough labeled data to build a supervised tagger. Table 3.2[1] shows some major languages with no or very limited annotated data. However, with the growth of multilingual websites, government documents and large archives of human translation of books, news and so forth, unannotated parallel data is becoming more widely available. We believe that we can base on parallel data to build a bridge between languages and then copy tagging information from one language to another. This approach also exploits the idea that the tag ambiguity in one language is likely to correspond to an unambiguous word in other languages. Consider the example, "*we can can a can*" as in Figure 3.1 and its Vietnamese translation "*Chung toi co the lam mot cai hop*". The ambiguous word "*can*" has several translations: "*co the*" (modal verb), "*lam*" (verb), "*cai hop*" (noun). Thus, the different translations help to disambiguate the syntactic category (POS tag) of word "*can*". Moreover, the underlining structure of a language can also be used to disambiguate. For example, in English, the word right after article is likely to be Noun or a capital word in German has a high chance to be Noun. Thus, the general idea of our Universal Tagger is exploiting the evidence from multiple languages via a parallel corpus.

[1]source : http://www.ethnologue.org/ethno_docs/distribution.asp?by=size

Language	**Source**	**#Tags**	**Accuracy**
Arabic	PADT/CoNLL07	21	96.1
Basque	Basque3LB/CoNLL07	64	89.3
Bulgarian	BTB/CoNLL06	54	95.7
Catalan	CESS-ECE/CoNLL07	54	98.5
Chinese	Penn ChineseTreebank	34	91.7
Chinese	Sinica/CoNLL07	294	87.5
Czech	PDT/CoNLL07	63	99.1
Danish	DDT/CoNLL06	25	96.2
Dutch	Alpino/CoNLL06	12	93
English	PennTreebank	45	96.7
French	FrenchTreebank	30	96.6
German	Tiger/CoNLL06	54	97.9
German	Negra	54	96.9
Greek	GDT/CoNLL07	38	97.2
Hungarian	Szeged/CoNLL07	43	94.5
Italian	ISST/CoNLL07	28	94.9
Japanese	Verbmobil/CoNLL06	80	98.3
Japanese	Kyoto4.0	42	97.4
Korean	Sejong	187	96.5
Portuguese	Floresta Sinta(c)tica/CoNLL06	22	96.9
Russian	SynTagRus-RNC	11	96.8
Slovene	SDT/CoNLL06	29	94.7
Spanish	Ancora-Cast3LB/CoNLL06	47	96.3
Swedish	Talbanken05/CoNLL06	41	93.6
Turkish	METU-Sabanci/CoNLL07	31	87.5
	Average		**95.2**

Table 3.1: Tagging Accuracy of many languages using TNT

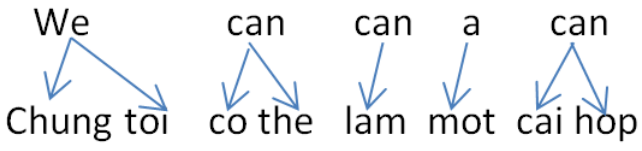

Figure 3.1: English - Vietnamese parallel sentence

Language	**Native Speaker(millions)**
Bengali	181
Javanese	84.6
Lahnda	78.3
Telugu	70
Vietnamese	69
Tamil	65.7
...	...

Table 3.2: Major languages with less annotated data

Tag	Description
NOUN	nouns
VERB	verbs
ADJ	adjectives
ADV	adverbs
PRON	pronouns
DET	determiner and articles
ADP	preposition and postpositions
NUM	numerals
CONJ	conjuctions
PRT	particles
.	punctuation marks
X	all other categories such as foreign word, abbreviation etc.

Table 3.3: 12 tags of the Universal tagset

3.2 Universal Tagset

The universal tagset is the tagset used for our universal tagger. One goal of universal tagger is to motivate the comparison between languages. Thus, universal tagset should be the common denominator of all languages. We adopt the work of (Petrov *et al.* 2012) on 12 universal tags which is described in Table 3.3. We believe that this basic 12 tags are shared among most languages of the world. The universal tagset benefits downstream applications that work across languages such as multilingual taggers, multilingual parsers and so forth. With the consensus tagset, the questions such as "Tagging language A is harder than language B" would become easier to answer. Moreover, the universal tagger exploits the idea of projecting tag information from a resource-rich language to a resource-poor one. Adopting the universal tagset means that we do not need to care about matching between different tagsets. In addition, we do not have any official tagset for many widely spoken

languages such as Telugu or Vietnamese. In this case, a universal tagset would be a good start.

Nevertheless, we acknowledge that we are loosing information using universal tagset. For example, all *VB, VBD, VBG, VBN, VBP, VBZ* tags from Penn treebank tagset are mapped to *VERB* of Universal tagset. However, this is the trade-off we need to make. Petrov *et al.* (2012) also provided the mapping from many other tagsets to Universal tagset. Table 3.1 lists all the available mapping from each language specific tagset to Universal tagset. In many languages the mapping is clear but in some languages it is not that obvious. Petrov *et al.* (2012) shows an example of Korean where the adjective is missing. They use stative verb to describe the property of objects, hence, stative verb is mapped to *ADJ* in universal tagset. The other example is the tagset used in French Treebank, there are no *NUM* tags. Numbers are tagged as adjective or nouns case by case.

Normally, the mapping is done manually by a linguistic expert. However, Zhang *et al.* (2012) also suggested methods to automatically map from language specific tagsets to the universal tagset.

3.3 Methodology

Here we describe our proposed tagger. The key idea is to maximize the amount of information gleaned from the source language, while limiting the amount of noise. Our approach contains 2 main steps (1) Building seed model (2) Apply self-training with revision. We describe the seed model and then explain how it is successively refined through self-training and revision.

3.3.1 Seed Model

The first step is to construct a seed tagger from directly-projected labels. Given a parallel corpus for a source and target language, Algorithm 6 provides a method for building an unsupervised tagger for the target language. In typical applications, the source language would be a resource-rich language having a tagger, while the target language would be resource-poor, lacking a tagger and large amounts of manually POS-labelled data.

Algorithm 6 Build seed model

1: Tag source side.
2: Word align the corpus with Giza++ and remove the many-to-1 mappings.
3: Project tags from source to target using the remaining 1-to-1 alignments.
4: Select the top n sentences based on sentence alignment score.
5: Estimate emission and transition probabilities.
6: Build seed tagger T.

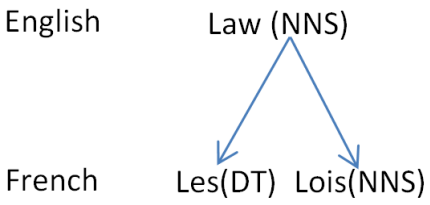

Figure 3.2: Many-to-one mapping case between English(law) and French(les lois)

Step 2 aligns words from source to target sentences. Words are aligned if they are translation of each other. There are cases where one word from source sentence is matched with exactly one word in the target sentence (one-to-one mapping) or one word in source sentence is mapped with a phrase in target sentence (many-to-one mapping). We eliminate many-to-one alignments (Step 2). Keeping these alignment would give more POS-tagged tokens for the target side, but also introduce noise. For example, suppose English and French were the source and target language, respectively. In this case alignments such as English *laws (NNS)* to French *les (DT) lois (NNS)* would be expected (Yarowsky and Ngai 2001) as shown in Figure 3.2. However, in step 3, where tags are projected from the source to target language, If we just copy tag information from English side *NNS* (common noun) to French side, both "*les*" and "*lois*" would be tagged as *NNS*. However, this is incorrect for "*les*" which must be *DT* (determiner).

We build a French tagger based on English–French parallel data from the Europarl Corpus (Koehn 2005). We want to know the accuracy and coverage of the tags obtained through direct projection. We thus compare these tags again French Melt POS tagger (Denis and Sagot 2009). This is a supervised French tagger built on French treebank (FTB) (Abeillé *et al.* 2003). They report the per-token accuracy is as high as 97.80% and thus, a good approximation for our experiment. Table 3.4 confirms that the one-to-one alignments indeed give higher accuracy but lower coverage than the many-to-one alignments. One-to-one alignment covers 68% of the token in French size (that is 32% is unaligned). However, if we use both one-to-one and many-to-one mapping, we cover 20% more of the tokens, but the accuracy is dropped by 10% (absolute). It's worth noting that, accuracy is only calculated from aligned tokens. Therefore, assuming that we have exactly 1000 tokens, if we used just one-to-one mapping, the number of correctly tagged tokens is $1000 \times 0.68 \times 0.78 = 530$ tokens. The number for additionally use many-to-one mapping is $1000 \times 0.88 \times 0.68 = 598$. That is we have $\frac{598-530}{1000} = 6.8\%$ more accurately tagged tokens. However, many-to-one introduces more noise (as mentioned in Figure 3.2), at this stage of the model we hypothesize that high-confidence tags are important, and hence we eliminate the many-to-1 alignments.

In Step 4, in an effort to again obtain higher quality target language tags from direct projection, we eliminate all but the top n sentences based on their alignment

Model	Coverage	Accuracy
Many-to-1 alignments	88%	68%
1-to-1 alignments	68%	78%
1-to-1 alignments: Top 60k sents	91%	80%

Table 3.4: Token coverage and accuracy of many-to-1 and 1-to-1 alignments, as well as the top 60k sentences based on alignment score for 1-to-1 alignments, using directly-projected labels only.

scores, as provided by the aligner via IBM model 3. We heuristically set this cutoff to 60k to balance the accuracy and size of the seed model[2]. Returning to our preliminary English–French experiments in Table 3.4, this process gives improvements in both accuracy and coverage. In the top 60k sentence, we just retain 1-to-1 alignment. The coverage is improved from 68% to 91% and the accuracy is also improved from 78% to 80%. We also considered using both one-to-one and many-to-one alignments for the top 60k sentences, but in preliminary experiments this did not perform as well, possibly due to the previously-observed problems with many-to-one alignments. Normally, the higher coverage, the lower accuracy and vice-versa, by simply ranking sentences and choose the first 60k, we have high coverage but in the same time, retain high accuracy. This is a *crucial step* in our proposed methods.

The step 5 which is for estimating transition and emission probability is based on the following intuition. The number of parameters for the emission probability $p(w|t)$ is $|V|\times|T|$ where w is word, t is tag, V is the vocabulary and T is the tag set. The transition probability $p(t_i|t_{i-1}t_{i-2})$, on the other hand, has only $|T|^3$ parameters for the trigram model we use. We use the universal tagset, thus, $|T|= 12$ but $|V|\approx 120k$ for French in a preliminary English-French experiment. Because of this huge difference in the number of parameters, in step 5, we employ different strategies to estimate the emission and transition probabilities. The emission probability is estimated from all 60k selected sentences. However, for the transition probability, which has less parameters, we again focus on "better" sentences, by estimating this probability from only those sentences that have

1. Token coverage $> 90\%$ (based on direct projection of from the source language)
2. Length > 4 tokens

These 2 criteria aim to identify longer, mostly-tagged sentences, which we hypothesize are particularly useful as training data. The underlying reason is that we want to estimate transition probabilities with as small bias as possible. For example, the first token of a sentence is likely to be subject and therefore, *Noun*. If we allow

[2]We considered values in the range 60–90k, but this choice had little impact on the accuracy of the model. Alternatively, we can set the threshold for sentence alignment score is 10^{-7}

short sentences (<= 3 tokens) when calculating trigram transition probability, the model will favor the distribution which start with *Noun*. This is undesirable for our model. In the same idea, a token missing inside a sentence disrupts the normal tag trigram distribution, thus, we reduce this by heuristically choosing only high coverage sentences ($> 90\%$) .

In the case of our preliminary English–French experiments, roughly 62% of the 60k selected sentences meet these two criteria and are used to estimate the transition probabilities. For unaligned words, we simply assign a random POS and very low probability, which does not substantially affect transition probability estimates.

In Step 6 we build a tagger by feeding the estimated emission and transition probabilities into the TNT tagger (Brants 2000), an implementation of a trigram HMM tagger.

3.3.2 Self training and revision

Up to this point we already have a tagger (seed model). In this section, we are going to improve this seed model by applying self-training with special revision. We exploit the large number of target language sentences available that have been partially tagged through direct projection, in order to build a more accurate tagger. Back to the preliminary English–French experiments, we just used 60k first sentence for the seed model, however, there are 1.9 millions other sentences. In this part, we are going to exploit the rest of data.

Algorithm 7 describes the process of self training and revision, and assumes that the parallel source–target corpus has been word aligned, with many-to-one alignments removed, and that the sentences are sorted by alignment score. In contrast to Algorithm 6, all sentences are used, not just the 60k sentences with the highest alignment scores.

Algorithm 7 Self training and revision

1: Divide target language sentences into blocks of n sentences.
2: Tag the first block with the seed tagger.
3: Revise the tagged block.
4: Train a new tagger on the tagged block.
5: Add the previous tagger's lexicon to the new tagger.
6: Use the new tagger to tag the next block.
7: Goto 3 and repeat until all blocks are tagged.

We believe that sentence alignment score might correspond to the difficulty of tagging. By sorting the sentences by alignment score, sentences which are more difficult to tag are tagged using a more mature model. Following Algorithm 6, we divide sentences into blocks of 60k. As mentioned before, many-to-one mapping introduce noise, therefore, we just remove it here.

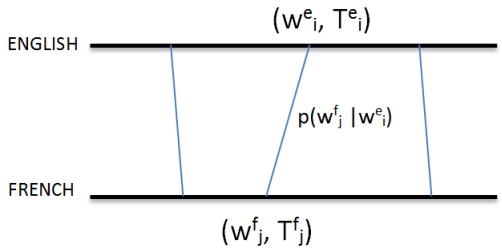

Figure 3.3: Alignment between source(English) and target(French) language

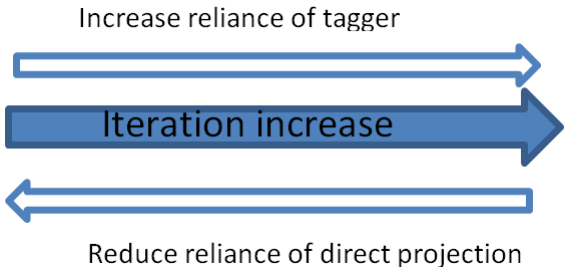

Figure 3.4: Reliance of tagger and direct projection label

In step 3 the tagged block is revised by comparing the tags from the tagger with those obtained through direct projection. Suppose source language word w_i^e is aligned with target language word w_j^f with probability $p(w_j^f|w_i^e)$, T_i^e is the tag for w_i^e using the tagger available for the source language, and T_j^f is the tag for w_j^f using the tagger learned for the target language as shown in Figure 3.3. If $p(w_j^f|w_i^e) > S$, where S is a threshold which we heuristically set to 0.7, we replace T_j^f by T_i^e. Self-training can suffer from over-fitting, in which errors in the original model are repeated and amplified in the new model (McClosky *et al.* 2006). To avoid this, we remove the tag of any token that the model is uncertain of, i.e., if $p(w_j^f|w_i^e) < S$ and $T_j^f \neq T_i^e$ then T_j^f = Null. So, on the target side, aligned words have a tag from direct projection or no tag, and unaligned words have a tag assigned by our model. By keeping the sentences in the order of difficulty to tag, the more we iterate, the more tokens are labeled by maturer model. It also means that, the more we iterate, we trust our model more and the direct projection less as shown in Figure 3.4.

Step 4 estimates the emission and transition probabilities as in Algorithm 6. In Step 5, emission probabilities for lexical items presented in the previous model, but missing from the current model, are added to the current model. Later models therefore take advantage of information from earlier models, and have wider coverage.

3.4 Experiment Detail

In this section, we are going to describe in detail our experiments. In the light of comparing with the state-of-the-art (Das and Petrov 2011), we experiment in the way that result would be comparable.

3.4.1 Source data

Using parallel data from Europarl (Koehn 2005) we apply our method to build taggers for the same eight target languages as Das and Petrov (2011) — Danish (da), Dutch (nl), German (de), Greek (el), Italian (it), Portuguese (pt), Spanish (es) and Swedish (sv) — with English as the source language. Das and Petrov (2011) choose these 8 languages simply because the amount of parallel data for these languages is numerous. Europarl is the corpus collected from European Parliament date back to 1996. At that time, official languages for European Union are only 11 languages. Aside from 8 languages we chose, 3 others are English (en), Finnish (fi) and French (fr). From 2004, European Union expand to 25 members and more languages are added. Thus, due to the long history, the amount of data for these 8 languages we chose are superior compare with most of other languages.

Our training data (Europarl) is a subset of the training data of Das and Petrov (2011), who also used the ODS United Nations dataset which we were unable to obtain. Less data and as will be shown, the result are comparable, actually make this work stronger than the state-of-the-art. The overview of Europarl for all 8 languages is given in the Table 3.5. Aside from Greek which has approximately 1.2 million sentences, other language pairs have approximate 1.9 million sentences.

Parallel Corpus (L1-L2)	**Sentences**	**L1 Words**	**English Words**
Danish-English	1,968,800	44,654,417	48,574,988
Dutch-English	1,997,775	50,602,994	49,469,373
German-English	1,920,209	44,548,491	47,818,827
Greek-English	1,235,976	32,031,068	31,929,703
Italian-English	1,909,115	47,402,927	49,666,692
Portuguese-English	1,960,407	49,147,826	49,216,896
Spanish-English	1,965,734	51,575,748	49,093,806
Swedish-English	1,862,234	41,508,712	45,703,795

Table 3.5: Overview of Europarl dataset for 8 languages

3.4.2 Word alignment

After collecting parallel data for each language, we are going to word align the data using the basic open source Giza++ to do the job. However before aligning, we

should pre-process the data and pre-compute files needed for alignment.

Tokenize the sentence

This step aim at separating each token by a space. It is not as easy as it appear. Model should be able to handle the punctuation well. For example, period "." which is used as sentence boundary should be separated but not in *"Mr. Paul"* or *"i.e"*. We use an open source *tokenizer.perl* tool from Giza++ package to tokenize both source and target file.

True casing

True casing help to choose the most proper casing for each words. For example, the word beginning sentence should be uppercase. However, defining sentence boundary might be tricky. The usual heuristic "." separate sentences is not always true. For example, "." in the composition cases such as "*i.e.*" or "*Mr.*" does not mark sentence ending. We also used an open source tool from Giza++ including two steps. (1) Train the model using *train-truecaser.perl* and (2) true casing using *truecase.perl*. The first steps will output the true casing model which is used in the second step.

Cleaning

Too long sentences are unreliable and not particularly suitable for our task. Therefore, we cut off sentence that longer than 80 words in both side of training data, using *clean-corpus-n.perl* also from Giza++ package. The above French-English parallel data from Europarl, approximately 43k (2.2%) sentences are cutoff.

Getting vocabulary and bitext file

Vocabulary file is needed to run Giza++. It lists the vocabulary and word's frequency in the training data. We must obtain vocabulary file for both source and target language. We use tool *plain2snt.out* from Giza++ package to build vocabulary for both source and target langauge at once.

```
plain2snt.out source target
```

The output will be `source.vcb, target.vcb, source_target.snt, target_source.snt` The first 2 files are vocabulary files for source and target language. The next 2 files are bitext files which represent parallel sentence as sequence of number. Each number is an id in the vocabulary file. Bitext files are also needed for word alignment.

Getting word classes

Word-class files are used just for IBM-model 4/HMM. Syntactic and semantic similar words are grouped into classes. We make use of *mkcls* tool. It's important that word class file name must comply with vocabulary file name.

```
mkcls -psource -Vsource.vcb.classes
mkcls -ptarget -Vtarget.vcb.classes
```

Word Alignment

Everything needed to run word alignment is ready. There are many possible parameters, however, we use the default setting for this experiment.

```
GIZA++ -S source.vcb -T target.vcb -C source_target.snt -o ALIGN
```

Sometime, co-occurrence file is needed. This file can easily acquire using *snt2cooc.out* tool. The output would have `ALIGN` prefix. There are many output files however, we particularly interested in (1) Lexical translation table file i.e. `ALIGN.t3.final`. (2) Alignment file i.e. `ALIGN.A3.final`. From these 2 files we will know the alignments and their confidence. Alignment is the most time consuming step in our whole process. To align each language pair in the Europarl corpus, it took us approximately a day on a eight core Intel Xeon 3.16 GHz CPU with 32 Gb Ram. The disadvantage of Giza++ is that, we can not achieve many-to-many mapping. That is, it always aligns from source to target, thus, just produce many-to-one and one-to-one mapping. We can fix this one by running Giza++ from other direction from target to source and merge two results. However, this step will double the running time. We will leave it for future work.

3.4.3 Direct tag projection

We tag the source file (English) with English Stanford POS Tagger (Toutanova *et al.* 2003a). This available tagger employs Penn treebank tagset which contain of 45 tags. Afterward, we use the mapping from (Petrov *et al.* 2012) to map into Universal Tagset (12 tags). Then tags are copied from source side (English) to target side (foreign language) using just one-to-one alignment.

3.4.4 Seed model and final model

We get sentence score from IBM model 3, that is, `ALIGN.A3.final`. Sentence score is used to rank sentence. The following is the sentence pair getting from English-French Europarl parallel data. English sentence *"resumption of the session"* is aligned with French sentence *"reprise de la session"* with sentence alignment score = 0.006578.

Language	Source	Number of Words
da	DDT/CoNLL06	94386
nl	Alpino/CoNLL06	203568
pt	Floresta/CoNLL06	206678
sv	Talbanken/CoNLL06	191467
el	GDT/CoNLL07	65419
it	ISST/CoNLL07	76295
es	Cast3LB/CoNLL06	89334
de	Tiger/CoNLL06	712332

Table 3.6: Size and source of annotated data

```
# Sentence pair (1) source length 4 target length 4 alignment score : 0.006578
reprise de la session
NULL ({ }) resumption ({ 1 }) of ({ 2 }) the ({ 3 }) session ({ 4 })
```

First 60k sentences are used to build seed model which is described in section (3.3.1). Then, we use seed model to build final model by applying self-training and revision, section 3.3.2.

3.4.5 Evaluation

We used the same per-token accuracy evaluation metric as the state-of-the-art (Das and Petrov 2011). Test data for all 8 languages (Danish, Dutch, German, Greek, Italian, Portuguese, Spanish, Swedish) are also the same. These test data which is mainly come from CoNLL-X and CoNLL-07 share tasks. Table 3.6 shows the source and size of test data for each language. We also use the mapping from (Petrov *et al.* 2012) to map each language into Universal Tagset.

3.5 Experimental Results

We experiment as described in previous section 3.4. Using parallel data from Europarl (Koehn 2005) we apply our method to build taggers for the same eight target languages as (Das and Petrov 2011) with English as the source language. Our training data (Europarl) is a subset of the training data of Das and Petrov (2011). The evaluation metric and test data are the same as that used by Das and Petrov. Our results are comparable to theirs, although our system is penalized by having less training data.

Table 3.7 shows results for our seed model, self training and revision, and the results reported by Das and Petrov. Self training and revision improves the accuracy for every language over the seed model, and gives an average improvement of roughly two percentage points. The average accuracy of self training and revision is on par

	da	nl	de	el	it	pt	es	sv	Avg.
Seed model	83.7	81.1	83.6	77.8	78.6	84.9	81.4	78.9	81.3
Self training + revision	**85.6**	**84.0**	**85.4**	80.4	81.4	86.3	83.3	**81.0**	83.4
(Das and Petrov 2011)	83.2	79.5	82.8	**82.5**	**86.8**	**87.9**	**84.2**	80.5	**83.4**
Prop. of unknown words	10.9	10.7	10.6	6.0	12.8	9.6	8.0	7.3	9.5

Table 3.7: Token-level POS tagging accuracy for our seed model, self training and revision, and the method of (Das and Petrov 2011). The best results on each language, and on average, are shown in bold.

with that reported by Das and Petrov. On individual languages, self training and revision and the method of Das and Petrov are split — each performs better on half of the cases. Interestingly, our method achieves higher accuracies on Germanic languages — the family of our source language, English — while Das and Petrov perform better on Romance languages. This might be because our model relies on alignments, which might be more accurate for more-related languages, whereas Das and Petrov additionally rely on label propagation.

The last row of Table 3.7 shows the percentage of unknown words for each language on the test data. On average, approximately 10% of the test data tokens are unknown. One way to improve the performance of our tagger might be to reduce the proportion of unknown words by using a larger training corpus, as Das and Petrov did. We should also consider corpus that cover wider topics. Language used in Europarl is formal and domain specific such as news, politic, economic etc. However, test data covers many more subjects such as fiction, sport etc. That might be the reason why despite Europarl is a very huge corpus, unknown word (OOV) rate is still high.

Compared to Das and Petrov (2011), our model performs poorest on Italian, in terms of percentage point difference in accuracy. Figure 3.5 shows accuracy, accuracy on known words, accuracy on unknown words, and proportion of known tokens for each iteration of our model for Italian. Iteration 0 is the seed model, and iteration 31 is the final model. Our model performs poorly on unknown words as indicated by the low accuracy on unknown words. The "overall accuracy" line is always between "known accuracy" and "unknown accuracy" line weighted by "proportion of known tokens" line. Thus, despite unknown words' accuracy drops drastically from 62% to 47%, an overall accuracy still (slightly) increases thank to increment on known accuracy words and proportion of known tokens.

The poor performance on unknown words is expected because we do not use any language-specific rules to handle this case. Moreover, for the final model, approximately 13% of the test data tokens are unknown. This is the highest rate compare to other languages as in Table 3.7. This also partially explain why performance is poor on Italian.

We examine the impact of self-training and revision over training iterations. We

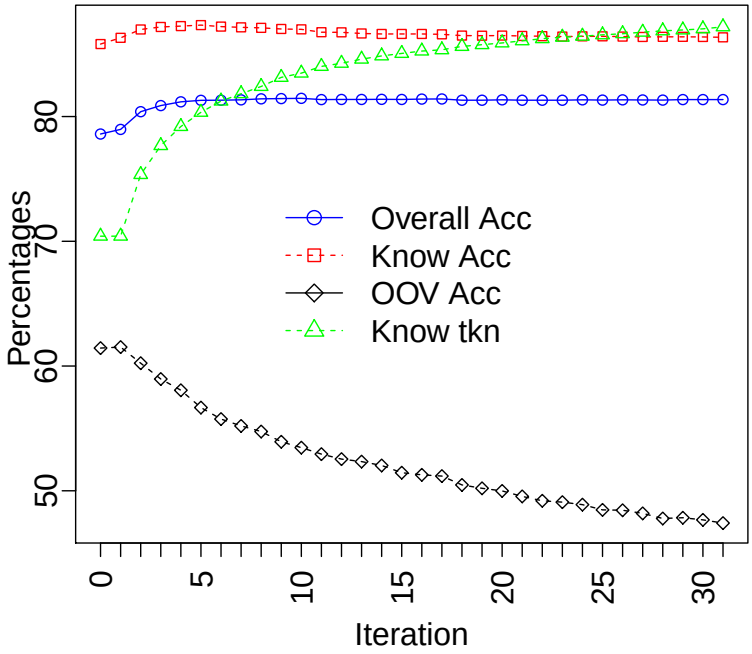

Figure 3.5: Overall accuracy, accuracy on known tokens, accuracy on unknown tokens, and proportion of known tokens for Italian.

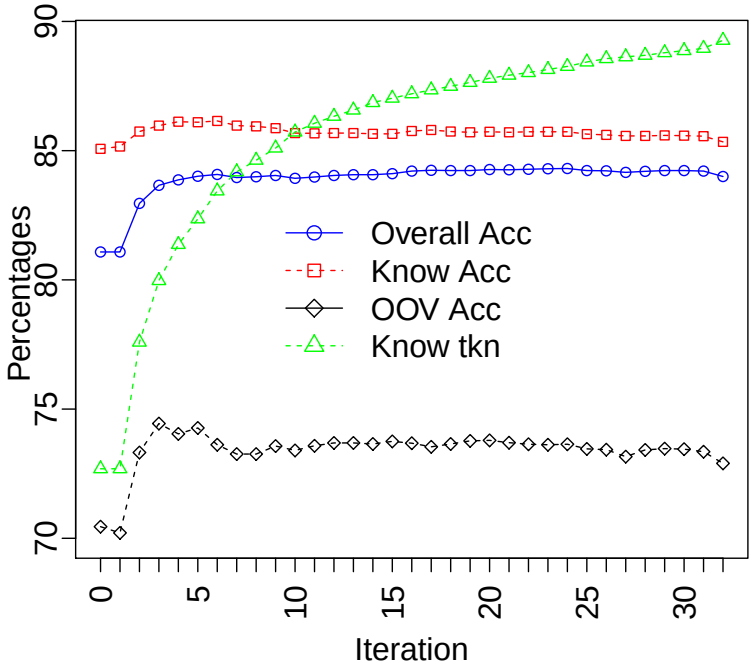

Figure 3.6: Overall accuracy, accuracy on known tokens, accuracy on unknown tokens, and proportion of known tokens for Dutch.

find that for all languages, accuracy rises quickly in the first 5–6 iterations, and then subsequently improves only slightly. We exemplify this in Figure 3.6 for Dutch, findings are similar for other languages. Although accuracy does not increase much in later iterations, they may still have some benefit as the vocabulary size continues to grow.

3.6 Summary of Contributions

We have proposed a method for unsupervised POS tagging (Universal tagger) that performs on par with the current state-of-the-art (Das and Petrov 2011), but is substantially less-sophisticated (specifically not requiring convex optimization or a feature-based HMM). The complexity for fully optimizing a convex function is $O(n^3)$ where n is size of data. This complexity is impractical because n is very big ($n \approx 2$ millions). Das and Petrov (2011) avoid this by just optimize for 10 iterations, instead of looping until converge. The complexity for each iteration is $O(n^2)$. So the overall complexity is $O(n^2)$. The complexity of our algorithm, on the other hand, is just $O(nlogn)$. This complexity is derived from sentence score sorting operation. The huge difference in complexity resulting in substantial running speed variance. We re-implemented label propagation from (Das and Petrov 2011). It took over a day to complete this step on an eight core Intel Xeon 3.16 GHz CPU with 32 Gb Ram, but only 15 minutes for our model. Moreover, unlike Das and Petrov (2011) who can not publish their code due to company license restriction, we made our code available for download[3] which hopefully will aid the development of multilingual natural language processing (NLP).

In future work we intend to consider using a larger training corpus to reduce the proportion of unknown tokens and improve accuracy. We can align source and target language in both directions, i.e. from source to target and from target to source language. Merging two alignment might yield a better one-to-one mapping set.

Given the improvements of our model over that of Das and Petrov on languages from the same family as source language, and the observation of Snyder *et al.* (2008) that a better tagger can be learned from a more-closely related language, we also plan to consider strategies for selecting more appropriate source language for a given target language.

Using our final model with unsupervised HMM inference methods might improve the final performance too, i.e. use our final model as the initial state for HMM, then experiment with different inference algorithms such as Expectation Maximization (EM), Variational Bayers (VB) or Gibbs sampling (GS). We in fact have tried EM, but it did not help. The overall performance dropped slightly. This might be because self-training with revision already found the local maximal point. However, VB or GS might help. Gao and Johnson (2008) compare EM, VB and GS for unsupervised

[3]https://code.google.com/p/universal-tagger/

English POS tagging. In many cases, GS outperformed other methods, thus we would like to try GS first for our model.

Chapter 4

Source Language Selection

Bilingual corpora offer a promising bridge between resource-rich and resource-poor languages, enabling the development of multilingual NLP technologies for a far wider range of languages. In the previous chapter, we used bilingual corpora to build a tagger for the target language. Its performance is on par with the state-of-the-art (Das and Petrov 2011). In this chapter we would like to further improve our tagger by investigating source language factors. English is often used as a source language, but it is not the only available resource-rich language, and another choice may have a dramatic effect on performance. Where multiple source languages are available, what should we do? How can we combine them? The relationship of source language selection and the universal tagger will be thoroughly considered in this chapter.

4.1 Introduction

Parallel texts are becoming increasingly available through sources such as multilingual websites, multilingual documents and large archives of translation memory from books, news. Not only the size but also the number of languages covered by parallel data is increasing. The era of English dominating one side of parallel texts is shifting to a far wider range of languages. Parallel data can be exploited to bridge languages, and in particular, transfer annotated information from a highly-resourced *source* language to a lesser-resourced *target* language, to build unsupervised POS taggers as demonstrated in Chapter 3.

One issue in building such a tagger is choosing the source language. English is commonly used, because parallel data which has English on one side is often most readily available. However, the appropriate source language might depend on the target language. Chapter 3 shows that taggers for languages which are in the same language family tree (Germanic) with source language (English) perform better than the state-of-the-art (Das and Petrov 2011). Snyder *et al.* (2008) suggested that performance of Slovene tagger improves 7.69% (absolute) when paired with Serbian, a

very closely related language, but only 1.3% when paired with English. Reddy and Sharoff (2011) and Hana *et al.* (2004) show that for closely related languages, the transition probabilities for an HMM tagger can be used interchangeably. This experience demonstrates that the choice of source language might have a drastic effect on target language tagger performance. Moreover, if parallel data for a target language with more than one source language is available, it might be possible to exploit this additional information; however, this issue has not been explored to date. Thus, in this chapter we investigate the problem of making a good choice of source language(s).

In this chapter we are going to build unsupervised POS taggers for m language pairs using the tagger from chapter 3. We identify features — derived from both monolingual and parallel corpora — that we could use to predict the best source language to build a tagger for a given target language. We show that choosing an appropriate source language can improve the accuracy of our state-of-the-art unsupervised POS tagging methodology, compared to using a single fixed source language. This prediction can be done based on features of the source and target language derived from monolingual corpora, although further improvements can be obtained using the features based on parallel corpora. We then show that even better accurate prediction can be obtained by incorporating information from multiple source languages.

Formally speaking, assuming that we need to build tagger for target language t, we have all possible source languages $s_1, s_2...s_n$. In this chapter, we are going to answer two questions: (1) what is the best source language s_i? (2) Can we do better by combining multiple source languages? To answer these two questions, we experiment as follows:

1. Pick n languages (Section 4.2)
2. Collect $n \times (n-1)$ parallel data sets (Section 4.2)
3. Define features (Section 4.3)
4. Using parallel data from each language pair, build the tagger (Section 4.4)
5. Build a source language predicting model based on the features (Section 4.5)
6. Combine multiple source languages (Section 4.6)

4.2 Collect parallel data

We would like to conduct experiments on a resource-poor target language, however, it would be much harder to evaluate. We instead experiment with the same $n = 9$ languages (English, Danish, Dutch, Portuguese, Swedish, Greek, Italian, German, Spanish). We use the JRC-Acquis corpus which provides parallel data for every pair of 22 European languages (Steinberger *et al.* 2006). We thus, extract a subset of

Language	No. of Texts	No. of Words ($\times 10^6$)
en	23545	55.5
da	23624	50.9
nl	23564	56.8
pt	23505	59.6
sv	20243	47.0
el	23184	55.9
it	23472	57.2
es	23573	62.1
de	23541	50.9

Table 4.1: The number of texts and words for each language considered in the JRC-Acquis corpus.

72 language pairs. It's worth noting that we consider $(x - y)$ and $(y - x)$ to be two different language pairs.

JRC-Acquis contains of EU legislation, rights, agreements, declarations etc that must be translated to all participating EU countries (currently 22). Thus, theoretically, each document would have a translation into 21 other languages. However, many documents for some languages are unavailable. Only documents that have translation into at least 10 other languages are included in the JRC-Acquis corpus.

To the best of our knowledge, JRC-Acquis is the biggest corpus providing parallel data for all of the language pairs we consider. Table 4.1 shows some monolingual statistics about each language. Table 4.2 shows the size of parallel data (number of sentences) for each language pair. It is clear that, parallel data in JRC-Acquis is balanced, that is, the size of each language pair is approximately the same.

	en	da	nl	pt	sv	el	it	es	de	Avg
en	-	1.00	1.13	1.12	1.06	0.79	1.12	1.12	1.14	1.06
da	1.00	-	1.16	1.14	1.10	0.83	1.14	1.14	1.16	1.10
nl	1.13	1.16	-	1.13	1.07	0.85	1.14	1.13	1.14	1.09
pt	1.12	1.14	1.13	-	1.05	0.84	1.13	1.13	1.12	1.08
sv	1.06	1.10	1.07	1.05	-	0.77	1.06	1.05	1.08	1.03
el	0.79	0.83	0.85	0.84	0.77	-	0.84	0.86	0.89	0.83
it	1.12	1.14	1.14	1.13	1.06	0.84	-	1.13	1.13	1.09
es	1.12	1.14	1.13	1.13	1.05	0.86	1.13	-	1.12	1.08
de	1.14	1.16	1.14	1.12	1.08	0.89	1.13	1.12	-	1.10

Table 4.2: JRC-Acquis corpus size ($\times 10^6$) for every language pair.

Intuitively, there is the question why we do not use the same Europarl parallel corpus as in chapter 3. Europarl provides parallel data which has English on one

side. We can always create other language pairs by using English as a pivot language. However, apart from the language pairs that involve English, all the other language pairs have more substantial coverage in JRC-Acquis. This means that Europarl is English-oriented and not particularly suitable for our experiment. Table 4.3 shows the size of each language pair (number of sentences) obtained from Europarl. The average data size for parallel data having English as the source language is double or triple compared to the other languages.

	en	da	nl	pt	sv	el	it	es	de	Avg
en	-	1.97	2.00	1.96	1.86	1.24	1.91	1.97	1.92	1.85
da	1.97	-	0.34	0.38	0.43	0.98	0.54	0.35	0.40	0.68
nl	2.00	0.34	-	0.39	0.44	1.05	0.53	0.39	0.42	0.69
pt	1.96	0.38	0.39	-	0.45	1.03	0.54	0.38	0.44	0.70
sv	1.86	0.43	0.44	0.45	-	0.97	0.63	0.45	0.48	0.71
el	1.24	0.98	1.05	1.03	0.97	-	1.11	1.01	1.06	1.06
it	1.91	0.54	0.53	0.54	0.63	1.11	-	0.54	0.59	0.80
es	1.97	0.35	0.39	0.38	0.45	1.01	0.54	-	0.43	0.69
de	1.92	0.40	0.42	0.44	0.48	1.06	0.59	0.43	-	0.72

Table 4.3: Europarl corpus size ($\times 10^6$) for every language pair. Where suitable bilingual data is not available, English is used as a pivot language to derived the other language pair.

4.3 Features

In this section, we consider factors that influence the choice of source language. We divide the features into two categories: *monolingual features* which exploit only monolingual data, and *bilingual features* which exploit parallel data.

4.3.1 Monolingual features

Morphological complexity

Morphologically rich languages introduce complexity when aligning parallel data because there is much greater ambiguity in alignment. Given the reliance of our approach on high quality alignments, morphological complexity is an important factor to consider. We can estimate morphological complexity by counting the number of unique tokens, i.e. the vocabulary size. In table 4.4, Voc. Size column displays vocabulary estimates for each language, assuming a corpus of a million words. This estimate uses the source side of the Acquis Corpus, although any monolingual corpus would suffice.

Language	Corpus Size		Voc. Size
	Acquis	Europarl	
en	-	-	14810
da	1000785	1968800	29867
nl	1132352	1997775	21316
pt	1121460	1960407	19333
sv	1061156	1862234	29403
el	792732	1235976	34992
it	1122016	1909115	19310
es	1117322	1965734	18496
de	1136452	1920209	29860

Table 4.4: Corpus size (number of tokens) and vocabulary size, for each language, with English as the source language.

Language relatedness

Our nine languages belong to three language families: Germanic (English, Danish, Dutch, Swedish, German); Romance (Portuguese, Italian, Spanish), and Baltic (Greek). Previously in chapter 3, our Universal Tagger performed better than the state-of-the-art on four languages which are in the same language family (Germanic) as the source language (English). Thus, language relatedness is an important factor to consider.

We quantify language relatedness using lexicostatistics on the Swadesh 200 word list (Meyer 1992). Table 4.5 shows examples of some meanings in English. This list was chosen by linguist Morris Swadesh by carefully taking into consideration cultural independence and attestation in many languages. The Swadesh word list is sometimes called a "universal" vocabulary because these meanings appear in the largest number of languages. The Swadesh list is important in lexico-statistics for studying language relatedness using the method of glottochronology.

I	louse	tooth
and	blood	know
all	bone	die
who	egg	give
father	animal	sun
one	tail	moon
two	ear	water
fish	eye	salt
dog	nose	stone

Table 4.5: Some meanings from the Swadesh word list

Lexicostatistics involves the judgment of linguist about whether a given pair of words are cognates or not. Two words are cognate if they evolved from the same ancestor. For example "*night*" (English), "*nuit*" (French) and *"Nacht"* (German) are cognate because they all derived from Proto-Indo-European word "*nokwts*"[1].

The relatedness of two languages is just the percentage of shared cognates in the word list. For example, Table 4.6 measures the relatedness between German and Spanish. Assume that for all 200 meanings, there are 90 "yes" in the *Cognate* column, the language relatedness of German and Spanish would be $\frac{90}{200} = 0.45$. Note that this measurement is symmetric. Two languages are considered close to each other if this value is high (close to 1). The postulation of language families is also partially based on this measurement. Languages that are close to each other are grouped into the same family.

No.	Meaning	German	Spanish	Cognate
1	all	alle	todo	no
2	and	und	y	no
3	father	Vater	padre	yes
4	animal	Tier	animal	no
..	..	..	..	..
200	I	Ich	yo	yes

Table 4.6: Language relatedness measure between German and Spanish

Meyer (1992) provides a table showing the language relatedness measurements for all 84 Indo-European languages. We thus extract a subset of 36 language pairs from this list. The extracted data is shown in Table 4.7[2].

	en	da	nl	de	el	it	pt	es	sv
en	-	0.593	0.593	0.578	0.162	0.247	0.24	0.24	0.589
da	0.593	-	0.663	0.707	0.183	0.263	0.25	0.25	0.874
nl	0.593	0.663	-	0.838	0.188	0.26	0.253	0.258	0.692
de	0.578	0.707	0.838	-	0.188	0.265	0.247	0.253	0.695
el	0.162	0.183	0.188	0.188	-	0.178	0.167	0.167	0.184
it	0.247	0.263	0.26	0.265	0.178	-	0.773	0.788	0.259
pt	0.24	0.25	0.253	0.247	0.167	0.773	-	0.874	0.258
es	0.24	0.25	0.258	0.253	0.167	0.788	0.874	-	0.253
sv	0.589	0.874	0.692	0.695	0.184	0.259	0.258	0.253	-

Table 4.7: Language relatedness measure for 9 languages

[1] http://en.wikipedia.org/wiki/Cognates

[2] for some language pairs, there are some meanings that do not present in these languages, thus, the final number is only calculated on a subset of 200 meanings

4.3.2 Bilingual features

Corpus size The most obvious feature is corpus size. The more data we have, the better. We count the number of parallel sentences in the corpus. Table 4.4 shows the corpus size for each language pair with English as the source side.

One-to-One alignment proportion We believe that one-to-one mapping is more meaningful for the task than many-to-one mapping. The intuition is that if there is only one possible way to copy a tag from the source language to the target language, we can be more confident about the mapping. The proportion of one-to-one mappings is calculated using a fixed number of parallel sentences (800k sentences) for all languages.

Sentence alignment score Sentence alignment scores are provided by the aligner from IBM Model 3. We used these scores to rank sentences in building a seed model as in chapter 3. This score has proven to be effective in choosing high quality sentences. Higher alignment scores might therefore correspond to a more accurate tagger. We use the average sentence alignment score for each language pair as a feature.

Lexical translation entropy We adopt the idea of translation model entropy from (Koehn *et al.* 2009). However, instead of scanning all possible sentence segmentations and calculating the phrase-based entropy, we employ a simpler method based on the lexical translation table. That is, the entropy for each lexical entry is calculated as

$$H(s) = -\sum_{t \in T} p(t|s) \times log_2 p(t|s)$$

where T is a possible translation of word s. For each language, we pick a fixed amount of text (1 million words) and calculate the average entropy for all words.

4.4 Build taggers

In this section we construct 72 taggers, using parallel data for 72 language pairs, and then evaluate the performance of each pair. We use the Universal tagger from chapter 3. Constructing a tagger for each language pair involves word alignment which is computationally expensive. Thus, we distribute the computation to four servers. Each server run multiple threads but it still took over 3 days for the whole process.

Our Universal tagger employs the consensus 12 Universal Tagset (Petrov *et al.*

Language	Source	Number of Words
en	WSJ/PennTB	1,289k
da	DDT/CoNLL06	94k
nl	Alpino/CoNLL06	203k
pt	Floresta/CoNLL06	206k
sv	Talbanken/CoNLL06	191k
el	GDT/CoNLL07	65k
it	ISST/CoNLL07	76k
es	Cast3LB/CoNLL06	89k
de	Tiger/CoNLL06	712k

Table 4.8: Size and source of annotated data

2012),[3] to avoid the problem of transliterating between different tagsets for different languages. Using this consensus tagset is crucial for enabling comparison across languages. The input for the Universal Tagger is a tagger for the source language $Tagger(s)$, along with parallel data $(s - t)$. The source language s is tagged using $Tagger(s)$, and then the tagged labels are projected to the target language side t. We rank and build a seed model T_0 on just the high scoring sentences. By applying self-training with revision, a series of new models is constructed $T_1, T_2, \ldots, T_m$. The output tagger for the target language is the last model $Tagger(t) = T_m$.

$Tagger(s)$ is trained from manually annotated data $Data(s)$ which is mainly derived from the CoNLL 06 and CoNLL 07 Shared Tasks. (Table 4.8 shows the source and size of annotated data for each language.) It is worth noticing that we only train on labeled data for the source language not for the target language. This means that training and testing data are always different for all language pairs. The annotated data was also used in the previous chapter 3 for evaluating the target tagger. We believe that the size of manually tagged corpus for each language is sufficient for building a reliable supervised POS tagger. Using the matching provided by Petrov *et al.* (2012), we map the individual tagsets to the Universal Tagset. We train a supervised POS tagger $Tagger(s)$ on the annotated data using the TNT tagger (Brants 2000). This data is also used for evaluating the target language tagger $Tagger(t)$.

We evaluate each $Tagger(t)$ using $Data(t)$, and report the results shown in Table 4.9. The average tagger performance for each source language is also given. It turns out that choosing Dutch (avg =72.78%) as the source language rather than English (avg=71.81%) gives the best overall performance. The tagger performance of each target language is much better than the baseline that always picks the most

[3]NOUN, VERB, ADJ, ADV, PRON (pronouns), DET (determiners and articles), ADP (prepositions and postpositions), NUM (numerals), CONJ (conjunctions), PRT (particles), "." (punctuation), and X (all other categories, e.g., foreign words, abbreviations).

SOURCE LANGUAGE \ TARGET LANGUAGE	en	da	nl	pt	sv	el	it	es	de	**Avg.**
en	-	76.17	72.97	79.57	**73.83**	50.38	72.20	75.37	73.95	71.81
da	55.73	-	53.28	50.53	66.08	34.13	46.03	50.34	53.90	51.25
nl	**75.70**	**76.31**	-	78.92	70.24	54.22	70.49	76.90	**79.47**	72.78
pt	72.40	69.49	63.07	-	66.67	61.82	**74.23**	80.50	64.70	69.11
sv	66.56	75.82	61.20	65.51	-	52.74	58.93	63.88	64.48	63.64
el	47.67	49.50	49.75	57.11	46.64	-	47.33	62.29	55.16	51.93
it	74.50	71.60	68.19	**84.50**	67.92	47.33	-	**81.80**	68.28	70.52
es	68.76	68.83	66.34	80.72	68.83	**62.29**	74.07	-	70.36	70.03
de	72.24	74.48	**76.54**	70.87	66.56	55.16	56.98	70.84	-	67.96
Baseline	30.28	23.27	24.28	24.53	26.35	24.00	25.09	21.98	26.50	

Table 4.9: Tagger accuracy for each source–target language pair. The best tagger for each target language is shown in bold.

frequent tag for each word.[4]

The Greek tagger performs poorly. From Table 4.4, Greek is the most morphologicaly complex language in this set, and has the smallest corpus size, two factors which partially explain why tagger performance for Greek is low regardless of whether Greek occupies the source or target language role.

From Table 4.9, it seems that taggers perform better if the source and target language are in the same language family. For example, the top four source languages for Danish are English, Dutch, Swedish and German, and the top two source languages for Portuguese are Italian and Spanish. This confirms the intuition in adding language relatedness features in section 4.3.

In chapter 3, we also used English as the source language to build taggers for the same eight other languages. The only difference between these two experiments is that in chapter 3 experiment, we used Europarl (Koehn 2005) data instead of JRC-Acquis. Table 4.10 compares the performance of each target language using English as the source language, for the two datasets. As mentioned before, Europarl is English oriented and not relevant for our experiment. Table 4.4 also compares the size of parallel data having English as the source language, for both corpora. Given that Europarl is much larger, higher performance is expected. However, Table 4.10 shows a strong correlation between the two experiments (Pearson's $r = 0.7$).[5]

This suggests that, if we had as much data as Europarl for every language pair (not just English), we would expect all numbers in Table 4.9 to improve substantially (not only the first row where English is the source language).

[4]For most of languages, the baseline is assigning "Noun" for all words.

[5]The common rule of thumb interpretation for Pearson-correlation is as follows: $|r| > 0.7$: very strong relationship, > 0.4 : strong relationship, > 0.3 : moderate relationship, > 0.2 : weak relationship, > 0.01: no or negligible relationship. Negative r means a negative relationship.

Language	JRC-Acquis	Europarl
da	76.2	85.6
nl	73.0	84.0
pt	79.6	86.3
sv	73.8	81.0
el	50.4	80.0
it	72.2	81.4
es	75.4	83.3
de	74.0	85.4
Average	71.8	83.4

Table 4.10: Accuracy on JRC-Acquis and Europarl using English as the source language

Features	r	r^2	Significance
Source Voc Size	-0.613	0.376	$***$
Target Voc Size	-0.202	0.041	$*$
Corpus Size	0.620	0.385	$***$
Language Relatedness	0.497	0.247	$***$
Sentence Alignment Score	0.492	0.242	$***$
One-to-one Mapping Proportion	0.745	0.556	$***$
Lexical Translation Entropy	-0.590	0.348	$***$

Table 4.11: Individual feature Pearson-correlation

4.5 Source language prediction

In this section, using features defined in section 4.3 and the tagger performance shown in Table 4.9, we build a model that can predict the performance of the target language tagger given a source language.

4.5.1 Individual feature correlation

Firstly, we want to determine the correlation of individual features with tagger performance. Table 4.11 shows the Pearson's correlation (r) and coefficient of determination (r^2) of each feature. The r^2 value can show the explanatory power, i.e., the extent to which the variance in tagger performance is "explained" by that factor. Significance shows the reliability of the calculation ($***$ means *p-value* < 0.001, $**$ means *p-value* < 0.01, $*$ means *p-value*< 0.05).

Surprisingly, the one-to-one mapping proportion is very strongly correlated with tagger performance ($r = 0.745$). The value of $r^2 = 0.556$ means that 55.6% of the variance in the tagger performance can be explained solely by this factor. The negative

		TARGET LANGUAGE								
		en	da	nl	pt	sv	el	it	es	de
SOURCE LANGUAGE	en	-	72.99	**77.76**	80.35	**76.42**	60.40	74.90	71.18	**75.91**
	da	61.07	-	63.65	61.69	71.00	50.85	57.08	48.97	62.18
	nl	68.10	72.59	-	70.55	69.56	60.03	65.21	63.47	72.75
	pt	**71.57**	68.96	70.42	-	69.12	64.77	**79.27**	**79.77**	69.55
	sv	63.64	**75.14**	64.32	61.86	-	52.22	56.36	53.24	63.71
	el	49.55	50.26	55.24	57.17	48.18	-	56.15	52.76	53.71
	it	71.16	69.58	71.10	**83.89**	69.35	65.26	-	77.30	70.16
	es	70.41	69.48	69.47	82.86	67.33	**65.91**	77.45	-	67.87
	de	60.90	69.88	67.73	62.86	64.50	55.60	59.07	54.00	-

Table 4.12: Predicted Tagger Accuracy for each source-target language pair. The best predicted tagger for each target language is in bold

correlation for the lexical translation entropy model is easy to understand because lower entropy is better. The source language vocabulary size is highly negatively correlated, but that strong relationship is not found for the target language. This suggests that the model is not affected much by the target language, but prefers the source language to be morphologically simple.

The corpus size factor is highly positively correlated too. This confirms the intuition that more data is better. This strong relationship, together with the negative morphological complexity factor, consolidates the explanation above about the poor performance of the tagger for Greek, where the availability of data is very limited, and where Greek has the richest morphology of any language considered.

4.5.2 Building a predictive model

In this experiment we are interested in building a model that can predict the performance of a target language tagger given a source language. We fit all features into a multiple linear regression model. The r^2 value improved drastically to 0.74, meaning that 74% of variance in tagger performance is explained by the factors we have identified.

We evaluate our model in a leave-one-out cross validation experiment. To build a predictive model for language t, we remove data in Table 4.9 associated with t and train the multiple linear regression model $model(t)$ on the remaining data. So, given source language s and $(s-t)$ parallel data sets, $model(t)$ outputs the predicted performance of the tagger trained on $(s-t)$ parallel data sets. The predicted accuracy for each language pair is given in Table 4.12. However, the correlation of the predicted value (Table 4.12) with the original value (Table 4.9) is very high ($r = 0.81$).

We also build another predictive model based solely on monolingual features (morphology complexity and language relatedness). The intuition here is that, if we only

Target language	All features	Monolingual features	Fixed	Oracle
en	pt (72.40)	**nl (75.70)**	**nl (75.70)**	nl (75.70)
da	sv (75.82)	en (76.17)	**nl (76.31)**	nl (76.31)
nl	**en (72.97)**	**en (72.97)**	-	de (76.54)
pt	**it (84.50)**	es (80.72)	nl (78.92)	it (84.50)
sv	**en (73.83)**	**en (73.83)**	nl (70.24)	en (73.83)
el	**es (62.29)**	en (50.38)	nl (54.22)	es (62.29)
it	**pt (74.23)**	es (74.07)	nl (70.49)	pt (74.23)
es	**pt (80.50)**	**pt (80.50)**	nl (76.90)	it (81.80)
de	en (73.95)	en (73.95)	**nl (79.47)**	nl (79.47)
Average	**74.50**	73.14	72.78	76.07

Table 4.13: Best source language prediction (and corresponding tagger performance) for models exploiting all features, only monolingual features, and a fixed source language, as well as an oracle model that always picks the best language. The best (non-oracle) source language and accuracy for each target language is shown in bold.

have monolingual data and we are planning to build a tagger for target language t, what parallel data would we want to collect first? This monolingual model also shows a high correlation with the original table ($r = 0.74$). If we only use language relatedness, the correlation is very weak ($r = 0.13$), showing that language relatedness on its own is not effective at predicting the best source language.

The predicted best source language for each target language t is the language predicted to produce the highest accuracy tagger. For example, from Table 4.12, if we want to build a tagger for Danish (da), we will choose Swedish (sv) as the source language. Table 4.13 shows the source language prediction from models exploiting all features, and only monolingual features. The Fixed model always chooses Dutch (nl) as the source language, because Dutch gives the highest average accuracy (Table 4.9). The Oracle model always picks the best language, and gives the upper bound for the predictive model as a point of comparison. As expected, the model exploiting all features achieves a higher average accuracy than the monolingual model, which nevertheless still outperforms Fixed. With respect to the oracle upperbound, and Fixed baseline, the error rate reduction for the monolingual and all features models is 10.9% and 52.3%, respectively, showing the effectiveness of using a predictive model.

4.6 Multiple Source Languages

In this section we combine information from multiple source languages to build a single target language tagger. We take a simple approach as shown in Figure 4.1. Each s_i is a tagged corpus for source language i. POS tags are then projected to the target language side t for each corpus. We merge all of these partially-tagged

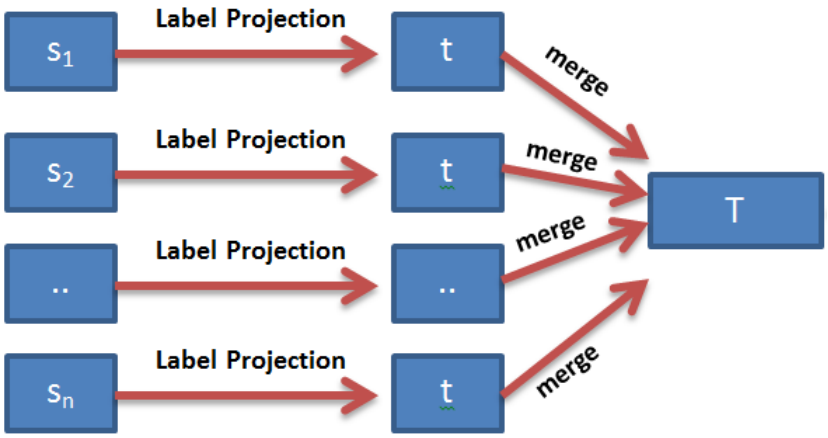

Figure 4.1: Combining multiple source language to produce single file

target language corpora (in which unaligned words are untagged) to form T. Because the JRC-Acquis corpus consists of translations of documents into multiple languages, in many cases the same target language sentence occurs in the parallel corpus for multiple source languages. In this preliminary approach to combining information from multiple source languages, we simply treat these as different target language sentences because the sentences are aligned with different source languages, they might contain different partial tag information.

We build the target language tagger from T by adapting the method from chapter 3 (Universal Tagger). The typical steps for this method are (1) tag the source language, (2) project labels from the source to target language, (3) build the seed model, and (4) apply self-training with revision to produce the final model. Here we simply start from step (3) and build the seed model from T.

In these experiments we assume that when building a tagger for a target language we have access to all other source languages. Table 4.14 shows accuracy when combining information from the 1-, 2-, 3-, to 7-best source languages (where the best source language is determined by an oracle). As more source languages are added, the average accuracy increases, although there is some variation for individual languages. Nevertheless, these findings show that the method described in chapter 3 is robust and can be substantially improved by combining information from multiple source languages. There is, however, a trade-off between accuracy and efficiency, with taggers built from multiple source languages generally being slower. The memory and running time is $O(nlogn)$ with the amount of data n. Table 4.14 also shows the running time for Portuguese on each combination. We run our experiment on a 16 cores Intel Xeon 2.53 GHz server with 24GB RAM.

Language	1 best	2 best	3 best	4 best	5 best	6 best	7 best
en	75.70	76.36	76.66	77.17	76.36	77.06	**78.16**
da	76.31	78.06	78.40	**83.35**	82.45	82.60	82.43
nl	76.54	76.82	76.17	76.03	80.00	**81.60**	81.45
pt	84.50	83.84	84.91	84.79	85.00	**85.46**	84.24
sv	73.83	74.33	74.65	74.49	74.10	74.51	**76.66**
el	62.29	66.90	**70.23**	67.93	67.22	67.03	67.69
it	74.23	77.28	**78.71**	78.47	78.47	78.03	76.05
es	81.80	81.89	82.53	**82.76**	82.13	82.21	82.64
de	**79.47**	79.35	79.28	78.79	77.92	77.88	77.35
Average	76.07	77.20	77.95	78.20	78.18	78.49	**78.52**
Run time(s)	352	444	741	1025	1648	2790	3297

Table 4.14: Tagger performance when combine multiple source language and performance of Portuguese languages measured in seconds. The best system are in bold

4.7 Summary of Contributions

In this chapter, we investigated the problem of choosing the best source language(s) to use in unsupervised multilingual POS tagging based on tag projection in parallel corpora. We have shown that our predictive model can select a source language — based on only monolingual features of the source and target languages — that improves tagger accuracy compared to choosing the single best (overall) source language. However, if parallel data is available, our predictive model is able to leverage this to select a more appropriate source language and obtain further improvements in accuracy. Finally, we showed that if multiple source languages are available, even better accuracy can be obtained by combining information from just those sources that are selected by our model. A synopsis of the process for building a tagger for language S is described in Figure 4.2. That is, if we do not have any parallel data, we would use a monolingual model to predict the best source language T and collect $S-T$ parallel data sets. If multiple parallel data sets are available and we have time, the best solution is just to combine all source languages to produce the single best tagger. If we do not have time, just combine n-best source languages in the order defined by *all features model* gave the comparable accuracy but stay fast.

In future work, we would like to apply the methods described in this paper for identifying "good" source languages for other multilingual NLP tasks which exploit parallel data to transfer annotations between languages, including grammar induction, parsing, and morphological analysis. We further intend to expand our experiments to consider more source and target languages. We also would like to investigate more methods for combining different source languages. Currently, we ignore cross-language sentences and concatenate them without any further processing. This resource can be very valuable since we have the POS information from all other lan-

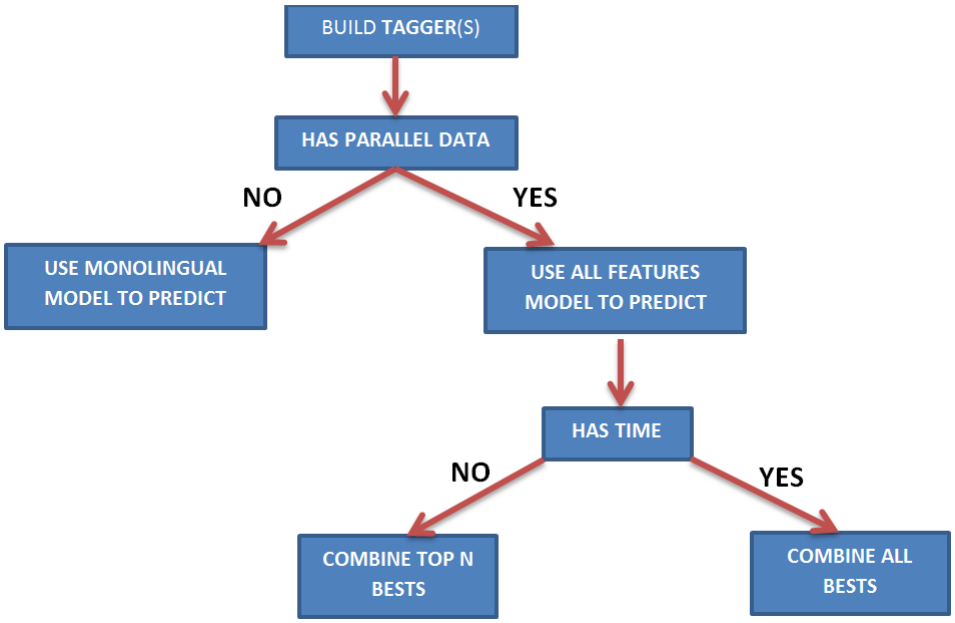

Figure 4.2: Instruction for building tagger for target language S

guages for each individual sentence. Thus, we can combine and produce more accurate tagged sentences which will serve as better training data for the POS tagger.

Chapter 5

Conclusions

POS tagging is an important task as it gives syntactic information and also helps to differentiate semantic information. There are three main challenges for POS tagging, (1) lack of manually annotated data which makes supervised approach impossible for resource-poor languages; (2) low performance on the traditional unsupervised approach which looks at each language separately; and (3) lack of tagset consensus among languages which is an obstacle for cross-language processing. In this thesis we tackle these challenges. We have successfully built an unsupervised multilingual POS tagger, but additionally, exploiting parallel data to copy tag information from resource-rich to resource-poor languages. In our tagger, we also employ the Consensus 12 Universal Tagset across languages which will resolve the third challenge.

Chapter 2 thoroughly reviewed approaches for POS tagging for both monolingual and multilingual taggers. Chapter 3 proposed the initial method for building Universal Tagger. That is, given a source language tagger and parallel data, we successfully construct a target language tagger. This tagger performs on par with the state-of-the-art system for the same 8 languages (Das and Petrov 2011). However, we use less data, simper methods – i.e. a huge difference in running speed, easier to replicate which actually makes our proposed method stronger. Unlike Das and Petrov (2011), we are able to publish the implementation[1] which might aid other researchers not only in multilingual POS tagging but also on other cross-languages NLP tasks such as parsing, grammar induction and so forth.

Out of 8 languages, the Universal Tagger performs better than the state-of-the-art on 4 Germanic languages which are in the same family as source language (English). It appears that choice of source language might substantially affect the target language tagger. This idea motivates our work in chapter 4. This chapter is the effort to further improve unsupervised multilingual POS tagger (Universal Tagger) accuracy by investigating the effect of choosing better source language(s). We found out that English is usually not the best source language. Just based on monolingual features,

[1] https://code.google.com/p/universal-tagger/

we are able to predict the best source language for a given target language. On average, the source language prediction gave better tagger performance than always fixing the source language. Even better accuracy can be obtain if we have parallel data. When multiple source languages are available, we shown that combining all of them even further improve tagging accuracy.

This thesis described a consensus works and thoroughly analysis of many aspects of building unsupervised multilingual POS tagger. Nevertheless, there are many points that could be improved. The program for future research is divided into 3 categories: (1) immediate works, which require less effort and can be done in few weeks; (2) near future works, which might require few months to complete; and (3) longer future works, which require years to complete.

Immediate works

1. Use a larger and more diverse traning corpus.

   The Universal Tagger we constructed in chapter 3 only uses the Europarl parallel corpus (Koehn 2005). Whereas the state-of-the-art (Das and Petrov 2011) additionally uses the ODS United Nations data set. Thus, we would like to acquire this corpus and add to the current Universal Tagger. This way, our model and the state-of-the-art will become fully comparable. Moreover, the current high rate of unknown word (OOV) when evaluating Universal Tagger against 8 languages suggested that enlarging the corpus might be the easiest and most promising way to improve performance.

2. Try bidirectional alignments

   Alignment is the one of the core modules of our Universal Tagger. Currently, we only employ one directional alignment from source to target language. It is possible to align in another direction from target to source language and then merge the results from these two alignments. In this way, we might have more and even better alignments. Actually, bidirectional alignment is widely used in statistical machine translation systems i.e. Moses (Koehn *et al.* 2007). We acknowledge this idea beforehand. However, alignment is the most time-consuming step in our model, bidirectional alignment doubles the running time, thus, we leave it for future work.

3. Build predicting model for many more languages

   This is a trivial extension of chapter 4. Currently, we are able to build a predicting model that predicts best source language given target language for 9 European languages, i.e. English, Dutch, Danish, German, Greek, Italian, Portuguese, Spanish, Swedish. We distinguish between two predicting models, that is, a *monolingual model* which just exploits monolingual data and an *all features model* which additionally exploits parallel data. We would like to extend this

to more languages, in a different strategy. First, for all 22 European languages in the JRC-Acquis Corpus (Steinberger *et al.* 2006), we have parallel data for each language pair, thus, we will build *all feature model* for better prediction. Second, we can further expand to more languages by just exploiting a *monolingual model* which only needs monolingual data. For example, we can easily build *monolingual model* for all languages presented in Wikipedia.

Near future works

1. Investigate methods for handling unknown word (OOV) case.

   As mentioned above, OOV rate stays high (nearly 10%) when evaluating the Universal Tagger on 8 languages. Currently, for unknown words (OOV), the model just uses the tag sequence to predict. For example, the tag sequence "*DET ADJ NOUN*" is commonly observed. So, if two previous words of unknown word are tagged as *DET* and *ADJ*, model infers *NOUN* for this word. The performance of this method on unknown words varies between languages but stays quite low i.e. 50-60% for Italian (Figure 3.5, page 40). Thus, given the high rate and currently low performance on OOV words, if we can incorporate other evidences to have better prediction for unknown words, we hope to improve the final performance.

2. Implement HMM inference algorithms

   We can use the Universal Tagger to initialize the first state of a Hidden Markov Model (HMM) and then use inference algorithms such as Expectation Maximization (EM), Variational Bayes (VB) or Gibbs Sampling (GS) to estimate the new set of parameters (emission probability and transition probability). Gao and Johnson (2008) compare EM, VB and GS for the same task of POS tagging, it seems that GS outperforms EM and VB. Thus, we would like to try GS first for our model.

3. Investigate more on combining multiple resources algorithms

   Chapter 4 shows that combining multiple source languages improves the overall accuracy. The current combining scheme is just concatenation, that is, ignore the existence of sentences that are shared among all languages. Thus, we would like to investigate methods that treat these sentences separately and propose a better combining scheme.

Longer future works

Our experience with parallel data, alignment, label projection, etc. will be the advantage for investigating similar unsupervised cross-language NLP tasks such as parsing, grammar induction, etc. The first task we would like to investigate is sentence parsing which is built upon POS information. However the ultimate goal for this

thread of work is building a framework for resource-poor languages, exploiting parallel data as the bridge between resource-rich and resource-poor languages. After being able to build another cross-lingual NLP applications, we would like to verify the source language predictive model proposed in Chapter 4 against these other applications.

Bibliography

Abeillé, Anne, Lionel Clément, and François Toussenel. 2003. Building a Treebank for French. In *Treebanks : Building and Using Parsed Corpora*, 165–188. Springer.

Baum, Leonard E., Ted Petrie, George Soules, and Norman Weiss. 1970. A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The Annals of Mathematical Statistics* 41.164–171.

Berg-Kirkpatrick, Taylor, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proceeding of HLT-NAACL*, 582–590. Association for Computational Linguistics.

Biemann, Chris. 2006a. Chinese whispers: an efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*, TextGraphs-1, 73–80. Association for Computational Linguistics.

——. 2006b. Unsupervised part-of-speech tagging employing efficient graph clustering. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, COLING ACL '06, 7–12. Association for Computational Linguistics.

Brants, Thorsten. 2000. TnT: A statistical part-of-speech tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing (ANLP '00)*, 224–231, Seattle, Washington, USA.

Brill, Eric. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics* 21.543–565.

Darroch, J. N., and D. Ratcliff. 1972. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics* 43.1470–1480.

Das, Dipanjan, and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, 600–609. Association for Computational Linguistics.

Denis, Pascal, and Benoît Sagot. 2009. Coupling an annotated corpus and a morphosyntactic lexicon for state-of-the-art POS tagging with less human effort. In *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation*, 721–736, Hong Kong, China.

——, and Benoît Sagot. 2012. Coupling an annotated corpus and a lexicon for state-of-the-art POS tagging. *Language Resources and Evaluation* 46.721–736.

Dien, Dinh, and Hoang Kiem. 2003. Pos-tagger for English-Vietnamese bilingual corpus. In *Proceedings of the HLT-NAACL 2003 Workshop on Building and using parallel texts: data driven machine translation and beyond - Volume 3*, HLT-NAACL-PARALLEL '03, 88–95. Association for Computational Linguistics.

Feldman, Anna, Jirka Hana, and Chris Brew. 2006. A cross-language approach to rapid creation of new morpho-syntactically annotated resources. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'06)*, 549–554, Genoa, Italy.

Gale, William. 1994. Good-Turing smoothing without tears. *Journal of Quantitative Linguistics* 2.

Gao, Jianfeng, and Mark Johnson. 2008. A comparison of Bayesian estimators for unsupervised hidden markov model POS taggers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, 344–352. Association for Computational Linguistics.

Gimenez, Jesus, and Lluis Marquez. 2004. SVMtool: A general POS tagger generator based on support vector machines. In *In Proceedings of the 4th International Conference on Language Resources and Evaluation*, 43–46.

Hajič, Jan, Alena Böhmová, Eva Hajičová, and Barbora Vidová-Hladká. 2000. The Prague Dependency Treebank: A Three-Level Annotation Scenario. In *Treebanks: Building and Using Parsed Corpora*, ed. by A. Abeillé, 103–127. Amsterdam:Kluwer.

Hana, Jiri, Anna Feldman, and Chris Brew. 2004. A resource-light approach to Russian morphology: Tagging Russian using Czech resources. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP '04)*, 222–229, Barcelona, Spain.

Hatori, Jun, Takuya Matsuzaki, Yusuke Miyao, and Junichi Tsujii. 2012. Incremental joint approach to word segmentation, POS tagging, and dependency parsing in Chinese. In *ACL (1)*, 1045–1053.

Johnson, Mark. 2007. Why doesnt EM find good HMM POS taggers. In *EMNLP*, 296–305.

Kilgarriff, Adam, Siva Reddy, Jan Pomiklek, and Avinesh PVS. 2010. A corpus factory for many languages. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Koehn, Philipp. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of the Tenth Machine Translation Summit (MT Summit X)*, 79–86, Phuket, Thailand. AAMT.

——, Alexandra Birch, and Ralf Steinberger. 2009. 462 Machine Translation Systems for Europe. In *Machine Translation Summit XII*, 65–72.

——, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, 177–180. Association for Computational Linguistics.

Manning, Christopher D. 2011. Part-of-speech tagging from 97 to 100 percent: is it time for some linguistics? In *Proceedings of the 12th International Conference on Computational linguistics and Intelligent Text Processing - Volume Part I*, CICLing'11, 171–189, Berlin, Heidelberg. Springer-Verlag.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19.313–330.

McClosky, David, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL '06)*, 152–159, New York, USA.

Meyer, Herbert A. 1992. An Indo-European classification: A lexicostatistical experiment. *Transactions of the American Philosophical Society* 82.825.

Petrov, Slav, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Ratnaparkhi, Adwait. 1996. A Maximum Entropy Model for Part-Of-Speech Tagging. In *Proceedings of the Empirical Methods in Natural Language Processing*, ed. by Eric Brill and Kenneth Church, 133–142.

Reddy, Raj, 1976. Beam search.

Reddy, Siva, and Serge Sharoff. 2011. Cross language POS taggers (and other tools) for Indian languages: An experiment with Kannada using Telugu resources. In *Proceedings of IJCNLP workshop on Cross Lingual Information Access: Computational Linguistics and the Information Need of Multilingual Societies. (CLIA 2011 at IJNCLP 2011)*, Chiang Mai, Thailand.

Reinsch, CH.H. 1967. Smoothing by spline functions. *Numerische Mathematik* 10.177–183.

Robertson, Stephen. 2004. Understanding inverse document frequency: On theoretical arguments for idf. *Journal of Documentation* 60.2004.

Ryan, Matthew S., and Graham R. Nudd. 1993. The Viterbi algorithm. Technical report, Coventry, UK, UK.

Schütze, Hinrich. 1995. Distributional part-of-speech tagging. In *Proceedings of the Seventh Conference at the European Chapter of the Association for Computational Linguistics*, EACL '95, 141–148, San Francisco, CA, USA. Association for Computational Linguistics.

Snyder, Benjamin, Tahira Naseem, Jacob Eisenstein, and Regina Barzilay. 2008. Unsupervised multilingual learning for POS tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '08)*, 1041–1050, Honolulu, Hawaii.

Søgaard, Anders. 2010. Simple semi-supervised training of part-of-speech taggers. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, 205–208. Association for Computational Linguistics.

——. 2011. Semisupervised condensed nearest neighbor for part-of-speech tagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, HLT '11, 48–52. Association for Computational Linguistics.

Steinberger, Ralf, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Toma Erjavec, and Dan Tufis. 2006. The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. In *In Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'2006)*, 2142–2147.

Toutanova, Kristina, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003a. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1 (NAACL '03)*, 173–180, Edmonton, Canada.

——, ——, ——, and ——. 2003b. Feature-rich part-of-speech tagging with a cyclic dependency network. In *IN PROCEEDINGS OF HLT-NAACL*, 252–259.

Tsuruoka, Yoshimasa, and Jun'ichi Tsujii. 2005. Bidirectional inference with the easiest-first strategy for tagging sequence data. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, 467–474.

Xi, Chenhai, and Rebecca Hwa. 2005. A backoff model for bootstrapping resources for non-English languages. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, 851–858. Association for Computational Linguistics.

Yarowsky, David, and Grace Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, NAACL '01, 1–8. Association for Computational Linguistics.

Yeh, Dashing. 1983. On incremental shift-reduce parsing. *BIT Numerical Mathematics* 23.36–48.

Yoshida, Kazuhiro, Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Tsujii. 2007. Ambiguous part-of-speech tagging for improving accuracy and domain portability of syntactic parsers. In *Proceedings of the 20th International Joint Conference on Artifical intelligence*, (IJCAI'07), 1783–1788, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Zhang, Yuan, Roi Reichart, Regina Barzilay, and Amir Globerson. 2012. Learning to map into a universal POS tagset. In *EMNLP-CoNLL*, 1368–1378.

Zhou, Zhi-Hua, and Ming Li. 2005. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering* 17.1529–1541.

Zweig, G., 1996. A forward-backward algorithm for inference in Bayesian networks and an empirical comparison with HMMs. Master's thesis, UC Berkeley CS Dept.

Appendix A

POS Tag	Description	Example
CC	coordinating conjunction	and
CD	cardinal number	1, third
DT	determiner	the
EX	existential there	there is
FW	foreign word	d'hoevre
IN	preposition/subordinating conjunction	in, of, like
JJ	adjective	green
JJR	adjective, comparative	greener
JJS	adjective, superlative	greenest
LS	list marker	1)
MD	modal	could, will
NN	noun, singular or mass	table
NNS	noun plural	tables
NNP	proper noun, singular	John
NNPS	proper noun, plural	Vikings
PDT	predeterminer	both the boys
POS	possessive ending	friend's
PRP	personal pronoun	I, he, it
PRP$	possessive pronoun	my, his
RB	adverb	however, usually
RBR	adverb, comparative	better
RBS	adverb, superlative	best
RP	particle	give up
TO	to	to go, to him
UH	interjection	uhhuhhuhh
VB	verb, base form	take
VBD	verb, past tense	took
VBG	verb, gerund/present participle	taking
VBN	verb, past participle	taken
VBP	verb, sing. present, non-3d	take
VBZ	verb, 3rd person sing. present	takes
WDT	wh-determiner	which
WP	wh-pronoun	who, what
WP$	possessive wh-pronoun	whose
WRB	wh-abverb	where, when

Table A.1: Penn Treebank Tagset (Marcus *et al.* 1993)