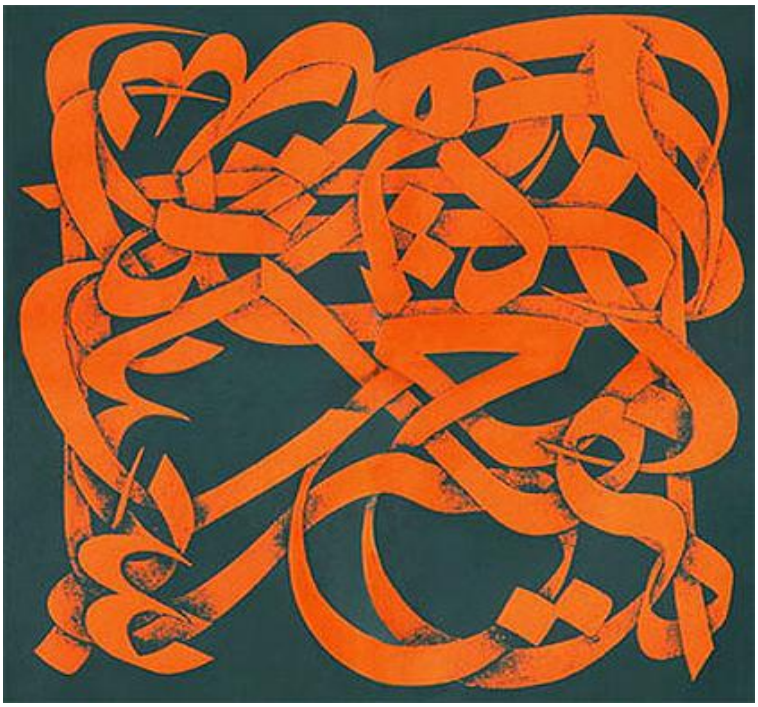
Computational Linguistics Project

Final Report

Two Novel Approaches for Persian Word Sense Disambiguation



By:

Bahareh Sarrafzadeh

Nikolay Yakovets

Instructor:

Professor Nick Cercone

Contents

Abstract

One of the primary challenges in Machine Translation is the *Word Sense Disambiguation* (WSD) problem. Word sense disambiguation is the ability to identify the meaning of words in context in a computational manner. Most of the approaches to the resolution of word ambiguity stem from the field of machine learning, ranging from methods with strong supervision, to syntactic and structural pattern recognition approaches.

As a matter of fact, statistical approaches require large amount of labeled resources as training datasets. There have been many statistical and knowledge-based approaches for solving WSD problem using *Classification Methods* in English. Since there are many available datasets and knowledge resources for this language, we witness several successful projects which have addressed WSD problem for English.

On the other hand, when it comes to Persian language, there is neither any semantically tagged corpus to aid Machine Learning approaches for Persian texts, nor any parallel corpora. Yet according to the ever-increasing development of Persian pages in Wikipedia, it can act as a parallel corpus for English-Persian texts.

In this project we first aimed to overcome the lack of knowledge resources and Text Understanding tools for Persian language by taking advantage of available English sense disambiguators, some special features – such as availability of articles in both languages – in Wikipedia and the newly developed lexical ontology, FarsNet, in order to address the WSD issue for Persian. Second, we tried one *direct* sense disambiguation approach for Persian to investigate both Cross Lingual and Knowledge based approaches for Persian WSD.

1 Introduction

Human language is ambiguous, so that many words can be interpreted in multiple ways depending on the context in which they occur. For instance, consider the following sentences:

- The *bank* cashed my check.
- We sat along the *bank* of the Tevere river.

The occurrences of the word *bank* in the two sentences clearly denote different meanings: “a financial institution” and “the land alongside or sloping down to a river or lake”, respectively. Unfortunately, the identification of the specific meaning that a word assumes in context is only apparently simple. While most of the time humans do not even think about the ambiguities of language, machines need to process unstructured textual information and transform them into data structures which must be analyzed in order to determine the underlying meaning. The computational identification of meaning for words in context is called *word sense disambiguation.*

WSD heavily relies on knowledge. In fact, the skeletal procedure of any WSD system can be summarized as follows: given a set of words (e.g., a sentence or a bag of words), a technique is applied which makes use of one or more sources of knowledge to associate the most appropriate senses with words in context. Knowledge sources can vary considerably from corpora (i.e., collections) of texts, either unlabeled or annotated with word senses, to more structured resources, such as machine-readable dictionaries, semantic networks, etc [1]. Without knowledge, it would be impossible for both humans and machines to identify the meaning, for example, of the above sentences. Unfortunately, the manual

creation of knowledge resources is an expensive and time consuming effort, which must be repeated every time the disambiguation scenario changes (e.g., in the presence of new domains, different languages, and even sense inventories). This is a fundamental problem which pervades the field of WSD, and is called the *knowledge acquisition bottleneck.*

This report is organized as follows: After a brief introduction to WSD task and different Knowledge resources, we review some related works in this field. Then we explain the Cross Lingual approaches in section 3. The novel approaches being utilized in this system and a comparison are discussed in section 4; which is followed by challenges for Persian and evaluation results. In the end a conclusion and future works are proposed.

1.1 Task Description

Word sense disambiguation is the ability to identify the meaning of words in context in a computational manner. WSD is usually performed on one or more texts (although in principle bags of words, i.e., collections of naturally occurring words, might be employed). If we disregard the punctuation, we can view a text T as a sequence of words (w_1, w_2, . . . , w_n), and we can formally describe WSD as the task of assigning the appropriate sense(s) to all or some of the words in T, that is, to identify a mapping A from words to senses such that $A(i) \subseteq Senses_D(w_i)$, where $Senses_D(w_i)$ is the set of senses encoded in a dictionary D for word w_i and $A(i)$ is that subset of the senses of w_i which are appropriate in the context T. The mapping A can assign more than one sense to each word $w_i \in T$, although typically only the most appropriate sense is selected.

1.2 WSD as a Classification Task

WSD can be viewed as a classification task: word senses are the *classes*, and an *automatic classification method* is used to assign each occurrence of a word to one or more classes based on the evidence from the *context* and from *external knowledge sources*. We can distinguish two variants [1] of the generic WSD task:

—*Lexical sample* (or *targeted WSD*), where a system is required to disambiguate a restricted set of target words usually occurring one per sentence. Supervised systems are typically employed in this setting, as they can be trained using a number of hand-labeled instances (*training set*) and then applied to classify a set of unlabeled examples (*test set*);

—*All-words WSD*, where systems are expected to disambiguate all open-class words in a text (i.e., nouns, verbs, adjectives, and adverbs). This task requires wide-coverage systems. Consequently, purely supervised systems can potentially suffer from the problem of *data sparseness*, as it is unlikely that a training set of adequate size is available which covers the full lexicon of the language of interest.

Although a targeted approach usually leads to a higher accuracy, we decided to use an All-words WSD method in order to increase the coverage of our system.

1.3 External Knowledge Resources

As it is mentioned before, Knowledge is a fundamental component of WSD. Knowledge sources provide data which are essential to associate senses with words. There are a variety of these resources: corpora of texts, either unlabeled or annotated with word senses, machine-readable dictionaries, thesauri, glossaries, ontologies, etc.

Among these resources, *Machine-readable dictionaries* (MRDs), have become a popular source of Knowledge for natural language processing. Currently the most utilized resource for word sense disambiguation in English is WordNet. WordNet is often considered one step beyond common MRDs, as it encodes a rich semantic network of concepts. For this reason it is usually defined as a *computational lexicon*. As we exploit WordNet in the process of English WSD, it is worthwhile to describe this resource in more details.

1.3.1 WordNet

WordNet is a computational lexicon of English based on psycholinguistic principles, created and maintained at Princeton University. While traditional dictionaries are arranged alphabetically, WordNet is arranged semantically, creating an electronic lexical database of nouns, verbs, adjectives, and adverbs [6]. It encodes concepts in terms of sets of synonyms (called *synsets*). WordNet can be considered as an electronic lexical database for English which is widely used in the Natural Language Processing (NLP) community for applications in Information Retrieval, Machine Translation and Word Sense Disambiguation [3]. Its latest version, WordNet 3.0, contains about 155,000 words organized in over 117,000 synsets. For example, the concept of *automobile* is expressed with the following synset:

{car, auto, automobile, machine, motorcar}

We can view a synset as a set of word senses all expressing (approximately) the same meaning. For example "*car*" can appear in different synsets corresponding to different meanings of this word.

The noun car has 5 senses (first 3 from tagged texts)

1. (598) car, auto, automobile, machine, motorcar -- (a motor vehicle with four wheels; usually propelled by an internal combustion engine; "he needs a car to get to work")
2. (24) car, railcar, railway car, railroad car -- (a wheeled vehicle adapted to the rails of railroad; "three cars had jumped the rails")
3. (1) cable car, car -- (a conveyance for passengers or freight on a cable railway; "they took a cable car to the top of the mountain")
4. car, gondola -- (the compartment that is suspended from an airship and that carries personnel and the cargo and the power plant)
5. car, elevator car -- (where passengers ride up and down; "the car was on the top floor")

As it can be seen in the example above, each synset is followed by a definition (*gloss*) and some examples.

2 Related Work

We can distinguish different approaches to WSD based on the amount of supervision and knowledge they demand. Hence we can classify different methods into 4 groups: Supervised, Unsupervised, Semi-

supervised and Knowledge-based. These groups are described in more details in the following subsections[1].

2.1 Supervised Approaches

These approaches use machine-learning techniques to learn a classifier from labeled training sets, that is, sets of examples encoded in terms of a number of features together with their appropriate sense label (or class); Usually, the classifier is concerned with a single word and performs a classification task in order to assign the appropriate sense to each instance of that word. The training set used to learn the classifier typically contains a set of examples in which a given target word is manually tagged with a sense from the sense inventory of a reference dictionary. Generally, supervised approaches to WSD have obtained better results than unsupervised methods. Different classification algorithms have been applied to WSD problem. As some examples we can refer to Decision Tree Learning, Naive Bayes Methods, Neural Networks, Example-based methods such as KNN, SVM.

Once an appropriate dataset which has word senses as class attributes is provided, Classification can be done easily by some available tools such as Weka. Since we did not find any suitable dataset for this particular application at the moment, we decided to try other approaches for performing WSD for English pages.

2.2 Unsupervised Approaches

These methods are based on unlabeled corpora, and do not exploit any manually sense-tagged corpus to provide a sense choice for a word in context. Unsupervised methods have the potential to overcome the knowledge acquisition bottleneck. These approaches to WSD are based on the idea that the same sense of a word will have similar neighboring words. They are able to induce word senses from input text by clustering word occurrences, and then classifying new occurrences into the induced clusters. They do not rely on labeled training text and, in their purest version, do not make use of any machine–readable resources like dictionaries, thesauri, ontologies, etc. While WSD is typically identified as a *sense labeling* task, that is, the explicit assignment of a sense label to a target word, unsupervised WSD performs *word sense discrimination*, that is, it aims to divide "the occurrences of a word into a number of classes by determining for any two occurrences whether they belong to the same sense or not" [1].

The main approaches to unsupervised WSD are methods based on Context Clustering and those based on Word Clustering. While in the former word senses are presented as first- or second-order context vectors, the latter aims at clustering words which are semantically similar and can thus convey a specific meaning.

2.3 Semi-supervised Approaches

Since Supervised methods need a huge amount of training data and Unsupervised methods suffer from low accuracy, Semi-supervised methods aim at using as little labeled data as possible while gain higher accuracy compared to pure unsupervised methods. *Bootstrapping Methods* are the best examples of these approaches which build a sense classifier with little training data, and thus overcome the main problems of supervision: the lack of annotated data and the data sparsity problem.

Bootstrapping usually starts from few annotated data *A*, a large corpus of unannotated data *U*, and a set of one or more basic classifiers. As a result of iterative applications of a bootstrapping algorithm, the annotated corpus *A* grows increasingly and the untagged data set *U* shrinks until some threshold is reached for the remaining examples in *U*. The small set of initial examples in *A* can be generated from hand-labeling [9] or from the automatic selection with the aid of accurate heuristics [10].

[1] This classification is based on [1] and the content is summarized to only focus on what we need in this project.

2.4 Knowledge-based Approaches

The objective of knowledge-based or dictionary-based WSD is to exploit knowledge resources (such as dictionaries, thesauri, ontologies, collocations, etc) to infer the senses of words in context. These methods usually have lower performance than their supervised alternatives, but they have the advantage of a wider coverage, thanks to the use of large-scale knowledge resources.
In the following subsection, one of the main knowledge-based techniques, which uses the overlap of sense definitions, will be discussed. Most approaches exploit information from WordNet in order to find similarities between words. An extension to the original Lesk algorithm will be introduced in the Direct Knowledge based section.

2.4.1 Overlap of Sense Definitions

This method which is also known as *Lesk Algorithm* was proposed by Michael Lesk in 1986. This algorithm identifies senses of words in context using definition (or gloss) overlap. It proceeds as follows:
_ From dictionary (WordNet), retrieve all sense definitions of the words in the context.
_ Determine the definition overlap for all possible pairwise sense combinations.
_ Choose senses that led to highest overlap. The following example (Figure 1.) will clarify this task:

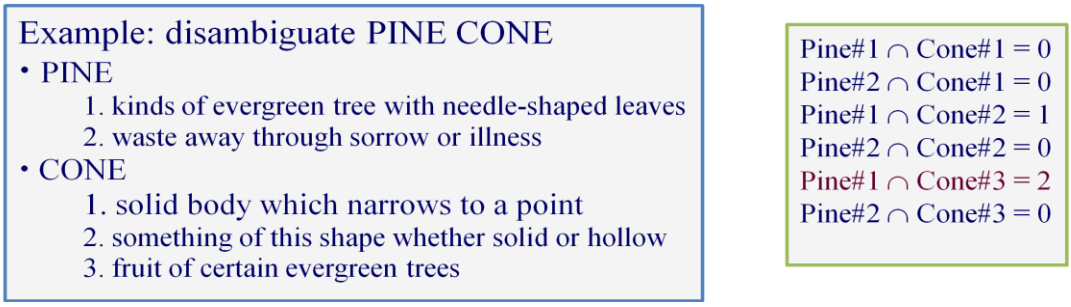


Figure 1. Lesk Algorithm - An example

There are different variations of Lesk algorithm. Apart from the one which mentioned above, we introduce two more variants here. The first one counts the number of words in common between the instance in which the target word occurs and its gloss, where each word count is weighted by its inverse document frequency (IDF). So in this variant we distinguish between content words and functions words (i.e. stop words, prepositions, etc.). The gloss with the highest number of words in common with the instance in which the target word occurs represents the sense assigned to the target word.

A second approach proceeded identically, except that it added example texts that WordNet provides to the glosses. These approaches have received accuracy of 16% and 23% respectively. In our project we refer to Lesk as the first variation mentioned above.

3 Cross-Lingual Approaches

Finally, we introduce an approach to disambiguation based on the evidence from translation information. The strategy consists of disambiguating target words by labelling them with the appropriate translation. The main idea behind this approach is that the plausible translations of a word in context restrict its possible senses to a subset [1]. For instance, the English word *"free"* can be translated to the Persian "آزاد"or "مجانی"depending on the context. However, this method does not necessarily perform a full disambiguation, as it is not uncommon that different meanings of the same

word have the same translation (e.g., both the senses of "*wing*" as an organ and as part of a building translate to the Persian "بال").
In recent studies, it has been found that approaches that use cross-lingual evidence for WSD attain state-of-the-art performance in all-words disambiguation. However, the main problem of these approaches lies in the knowledge acquisition bottleneck: there is a lack of parallel corpora for several languages – including Persian - which can potentially be relieved by collecting corpora on the Web. To overcome this problem, we utilized Wikipedia pages in both Persian and English languages.
Since there is a wide variety of disambiguation systems for English texts as well as desired knowledge-based lexical resources such as WordNet, we designed a novel approach to exploit the results of sense disambiguation for English words in order to assign appropriate senses to Persian words.
To this end, we utilized Wikipedia pages which are available in both languages and the newly developed Persian version of WordNet, FarsNet. Before describing our approach, we briefly introduce FarsNet.

3.1 FarsNet

FarsNet is an ongoing project to develop a lexical ontology to cover Persian words and phrases. It is designed to contain a Persian WordNet in its first phase and grow to cover verbs' argument structures (thematic roles and their selectional restrictions) in its second phase. Thus FarsNet consists of two main components:

- Persian WordNet
- Persian Net of verb frames (PeVNet)

The first release of FarsNet 1.0 which just contains Persian WordNet will be available shortly. It contains lexical, syntactic and semantic knowledge about more than 15000 Persian words and phrases organized in about 10000 synsets of nouns, adjectives and verbs. The included words and phrases are selected according to BalkaNet base concepts and the most frequent Persian words and phrases in our corpora. FarsNet 1.0 relates synsets in each POS category by the set of WordNet 2.1 relations. In the current version inter-POS relations are not implemented.
FarsNet also has inter-lingual relations connecting Persian synsets to English ones (in Princeton WordNet 3.0). Persian WordNet goes closely in the lines and principles of Major WordNets such as Princeton WordNet, EuroWordNet and BalkaNet to maximize its compatibility to these WordNets and to be connected to the other WordNets in the world to enable cross lingual tasks such as Machine Translation, multilingual IR and developing multilingual dictionaries and thesauri.

4 System Introduction

This system exploits two different approaches to assign sense tags to Persian words in some input Wikipedia pages. First the WSD task has been performed using a Cross Lingual approach. Then for each tagged word in a sentence we try to re-tag it using a direct knowledge approach. These approaches will be discussed in the following sections. A comparison is proposed in the third subsection .

4.1 Cross Lingual Approach: Persian WSD using tagged English words

This approach consists of two separate phases. In the first phase we utilize an English sense disambiguator to assign sense tags to words appearing in English Wikipedia pages. In the second phase we transfer these senses to Persian words in corresponding Persian Wikipedia pages. Please note that these two phases are completely distinct. Hence we can consider the first phase as a black box and we can use different English WSD systems here.

Three main components of this approach are described in the following sections. Figure 2 indicates the system's architecture for the Cross Lingual section.

4.1.1 English Sense Disambiguation

As it is mentioned, we can utilize different English Sense Disambiguator systems for this phase. First we tried to find some appropriate datasets which are specialized for the sense disambiguation application. Such dataset contains different features for describing words as attributes and word senses as class attributes. We can group these features as [1]:

> —*local features*, which represent the local context of a word usage, that is, features of a small number of words surrounding the target word, including part-of-speech tags, word forms, positions with respect to the target word, etc.;
> —*topical features*, which—in contrast to local features—define the general topic of a text or discourse, thus representing more general contexts (e.g., a window of words, a sentence, a phrase, a paragraph, etc.), usually as bags of words;
> —*syntactic features*, representing syntactic cues and argument-head relations between the target word and other words within the same sentence (note that these words might be outside the local context);
> —*semantic features*, representing semantic information, such as previously established senses of words in context, domain indicators, etc.

Based on this set of features, each word occurrence (usually within a sentence) can be converted to a feature vector [1].
Since no freely available dataset with these desired characteristics was found, we decided to exploit another disambiguation system for the English part. After testing several available sense disambiguator systems, we finally decided to use SenseRelate for the English WSD phase. This application is described in the following subsection.

4.1.1.1 SenseRelate

WordNet::SenseRelate::AllWords is freely available as a Perl-based application that uses WordNet to perform knowledge based word sense disambiguation. For every content word which is known to WordNet, it assigns a sense that is most related to senses of a corresponding surrounding words.
SenseRelate processes a given text sentence by sentence. For each sentence, it proceeds from left to right, centering each content word in a balanced window of context whose size is determined by the user. All senses of the centered word are fetched from WordNet and measured pairwise for similarity relative to the possible senses of surrounding words. The sense with the highest similarity is chosen to be the sense of that word. Then application considers next content word and continue until it reaches the end of the given sentence.
SenseRelate allows a user to specify a wide range of settings to control the desired disambiguation. Those range from selecting a format of input text to specifying a context window size used in disambiguation.
As an input to SenseRelate we fed plain untagged text of English Wikipedia pages that was carefully preprocessed according to application's preconditions. We also provided custom tweaked Stop Word list that is more extensive that the one that came bundled with the application.
SenseRelate also allows for a wide range of sense similarity and relatedness measures to be chosen. After careful consideration and experimenting we determined that measure that uses gloss overlaps (Lesk) coupled with window size of 5 led to most accurate disambiguation.

As an example, the output of this system for the given sentence below is as follows:

a bank is a financial institution licensed by a government

a : stopword
bank#n#2: a financial institution that accepts deposits and channels the money into lending activities; "he cashed a check at the bank"; "that bank holds the mortgage on my home"
is#v#1: have the quality of being; (copula, used with an adjective or a predicate noun); "John is rich"; "This is not a good answer"
a : stopword
financial_institution#n#1: an institution (public or private) that collects funds (from the public or other institutions) and invests them in financial assets
licensed#a#1: given official approval to act; "an accredited college"; "commissioned broker"; "licensed pharmacist"; "authorized representative"
by : stopword
a : stopword
government#n#1: the organization that is the governing authority of a political unit; "the government reduced taxes"; "the matter was referred to higher authorities"

As it can be seen in this example, the parts in blue indicate the tagged English words which are following this format:

word#part of speech#sense number in WordNet

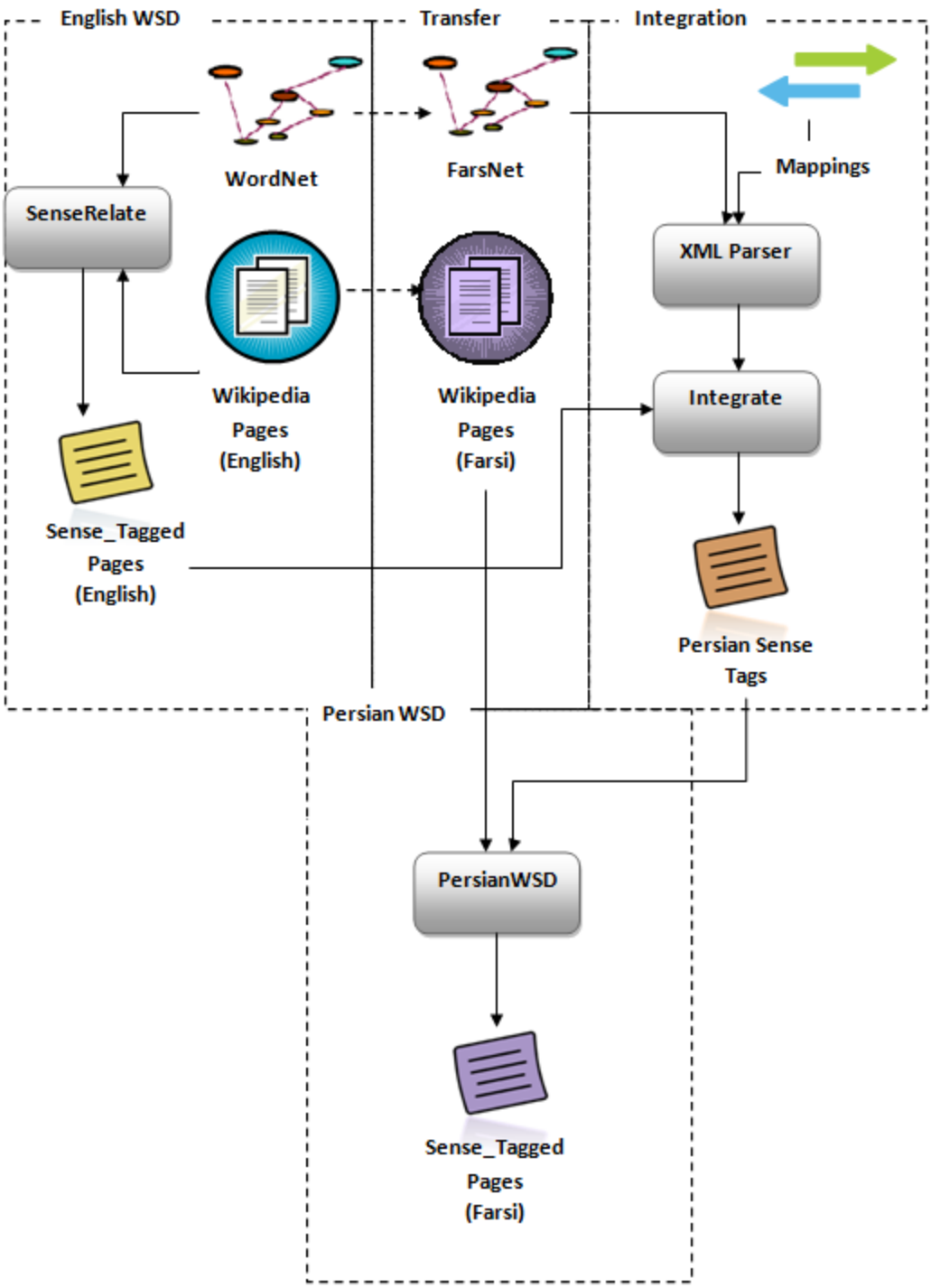


Figure 2. System Architecture for the Cross-Lingual approach

4.1.2 English to Persian Transfer

Running SenseRelate application for some Wikipedia pages, we have all English words tagged with word sense labels. Each of these sense labels corresponds to a synset in WordNet containing that word in a particular sense. In order to take advantage of these English tags for assigning appropriate senses to Persian words, first we need to transfer these synsets from English to Persian.

Interestingly, the Persian version of WordNet which is called FarsNet [2] will be available by Jan 2010. However, we had this chance to access the mappings between each synset in WordNet to its equivalent synset in FarsNet.

Exploiting these mappings, we match each WordNet synset which is assigned to a word in an English Wikipedia page to its corresponding synset in FarsNet. For this part, we developed a Perl-based XML-Parser and we integrate the results into the output provided by SenseRelate.

Along with transferring sense, we also need to transfer the Wikipedia pages from English to Persian. Here, we choose the pages which are available both in English and Farsi. Hence we can work with the pages describing the same title in Persian.

4.1.3 Assigning Sense Tags to Persian words

There are two different heuristics for assigning senses [1]:

—*one sense per collocation*: nearby words strongly and consistently contribute to determine the sense of a word, based on their relative distance, order, and syntactic relationship;

—*one sense per discourse*: a word is consistently referred with the same sense within any given discourse or document.

If there were available parallel corpora for English – Persian texts, we could align sentences from both languages and we would assign the same sense as the English word to its translation appearing in the aligned Persian sentence. In this case, we would get a very high accuracy, while our system would be limited to this specific type of corpora. Hence we did not use the first heuristic in our system.

Therefore, we decided to utilize Wikipedia pages which are available in both English and Farsi. Although the Farsi pages are not the direct translation of English pages, the context is the same for all corresponding pages, which implies many common words appear in both pages. Consequently, we can assume the domain-specific words appear with the similar senses in both languages.

Based on this hypothesis, for each matched synset in FarsNet which contains a set of Persian synonym words, we find all these words in the Persian text and we assign the same sense as the English label to them. As an example consider the sample output in the 4.1.1 section. SenseRelate assigned the second sense of noun “bank” to this word. The Persian equivalent noun (i.e. “بانک”) has six different senses. Among them we select the sense which is mapped to the second sense of word “bank” in WordNet and we assign this sense from FarsNet to “بانک”.

There are 3 possible scenarios:

(1) An English word has more than one sense, while the equivalent Persian word only has one sense. So, SenseRelate disambiguates the senses for this English word, and the equivalent Persian word doesn't need to be disambiguated. For example “*free*” in English is a polysemic word which can mean both “*able to act at will”* and “*costing nothing*”, while we have different words for these senses in Persian (“آزاد” and “مجانی” respectively). In this case we are confident that the transferred sense must be the correct sense for the Persian word. Because the Persian words are not ambiguous, so they will receive an accurate tag.

(2) Both the English and the Persian words have more than one senses, so as their contexts are the same, the senses should be the same. In this case we use FarsNet for the mapping between WordNet and the Persian synsets in FarsNet. For example the word "*branch*" in English and its Persian equivalent "شاخه" both are polysemic with similar set of senses. So, for example, if SenseRelate assigned the 5th sense (i.e "*a stream or river connected to a larger one*") of this word to its occurrence in an English sentence, the mapped synset in FarsNet would also correspond to this sense of the Persian "شاخه".

(3) The worst case happens when an English word only has one sense, while the Persian equivalent has more than one. In this case, as the context of both texts are the same, the Persian word is more likely to occur with the same sense as the English word. For example the noun "*Milk*" in English has only one meaning, while its translation in Farsi (i.e. "شیر") has three completely different meanings: Milk, lion and (water) tap. However, since SenseRelate assigns a synset with this gloss "a white nutritious liquid secreted by mammals and used as food by human beings" to this word, the first sense will be selected for "شیر".

In summary, for all these 3 possible scenarios we utilize the mappings from WordNet synsets to FarsNet ones. However, the first case usually leads to more accurate results and the third one has the lowest accuracy. Yet, when it comes to domain-specific words, all three cases will result in a high precision rate.

4.2 Direct Knowledge based approach: Applying Extended Lesk to Persian WSD

As it is mentioned Farsi Wikipedia pages are not the direct translation of their English counterparts. Hence, using the described Cross Lingual approach, we limit the system coverage to only those words which appear in both texts. In other words, even though we utilize an all-words WSD method for English, we transfer only some of these tags to Persian words. What is more, since English and Persian sentences are not aligned, for each English tagged word, we assign the same equivalent Persian sense tag regardless of the sentence this Persian word is occurred in.

Therefore, we can improve both accuracy and coverage of our system by utilizing some NLP and Knowledge-based approaches in the Persian WSD phase and integrate it with the senses transferred from the English phase. One of the first candidates for such knowledge based methods is applying the Lesk algorithm directly to the Persian WSD phase.

Thanks to the newly developed FarsNet, Lesk method (gloss overlap) is also applicable to Persian texts as well. Since this lexical ontology provides us with the glosses for each word sense, we are able to overlap these glosses for some neighbouring or co-occuring words (for example words appearing in the same sentence) and disambiguate their senses based on the senses with the maximum gloss intersection. Due to the fact that FarsNet has not been officially released, this method is not applied to Persian sense disambiguation yet. Hence, we were not certain how well this approach will perform for this purpose and that whether or not we still need transferring senses from English to Persian. These approaches (cross-lingual disambiguation and Lesk algorithm as an instance of a knowledge based method) can be complementary. Therefore it was worthwhile to investigate this idea further. Before describing the utilized knowledge based approach, we will have a brief introduction to word similarity measures. Figure 3 shows the organization of this implemented method.

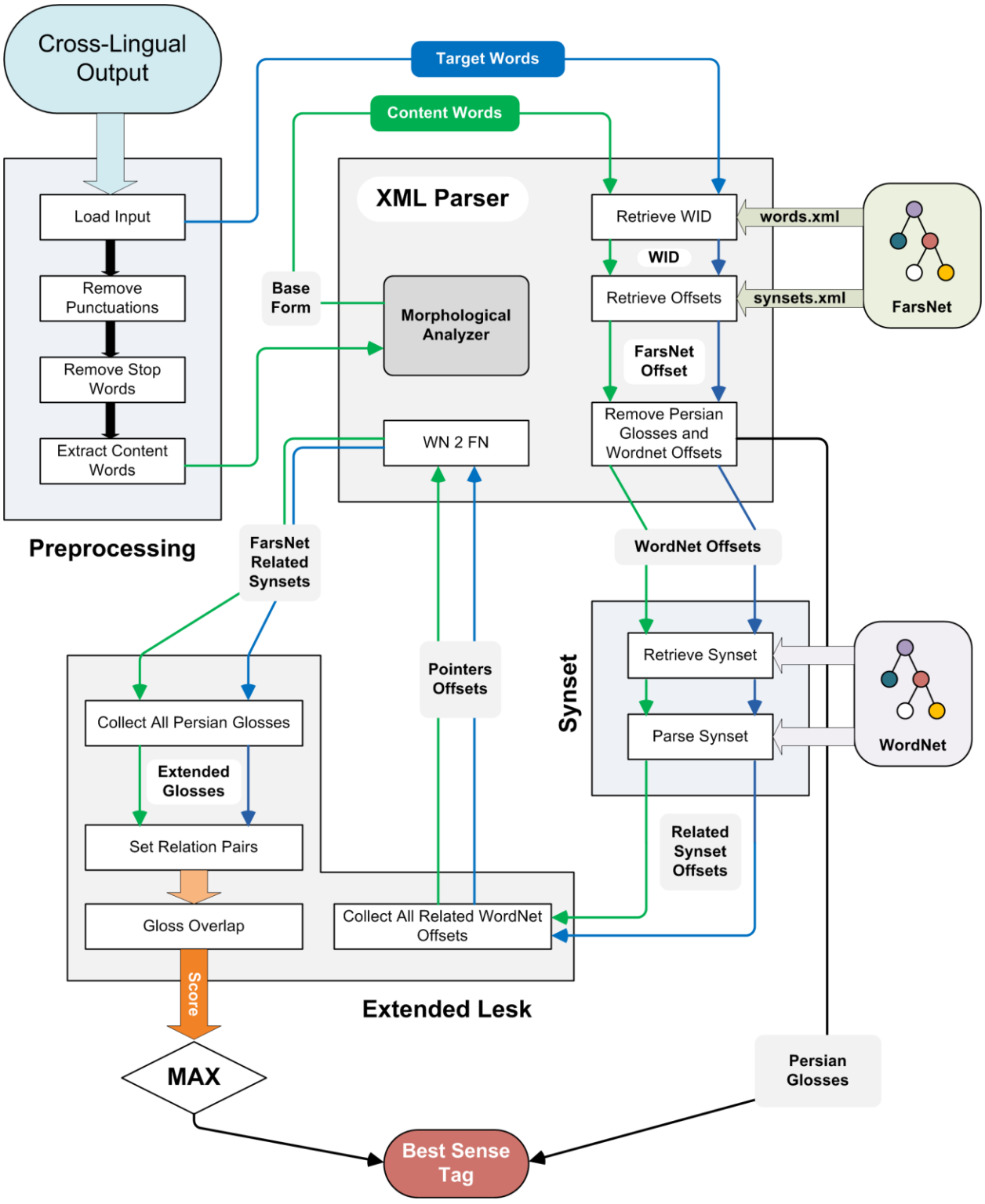


Figure 3. System Architecture for the Extended-based method

4.2.1 Semantic Similarities for Words

There are a variety of similarity measures to determine how much related two words are. First, we thought about building N-gram profiles for these sentences and compare them with some available measures such as Euclidean distance or Out of Place. However, since we don't have enough data – relevant sentences for each sense of a word – this statistical method will not be applicable.

Another method used to resolve WSD is the Lesk algorithm. The performance of this algorithm is connected with the similarity relatedness between all words in the text, so the success rate of WSD should increase as the similarity measure's performance gets better.

We, as human beings are usually able to tell if one word is more similar to a given word than another. For example, most would agree that the automotive senses of *car* and *tire* are related while *car* and *tree* are not.

There are mainly two approaches to semantic similarity [4]. In the first approach a large corpus is used to provide statistical data in order to estimate a score of semantic similarity. Second approach exploits the relations and the hierarchy of a thesaurus, which is generally a hand-crafted lexical database such as WordNet. There are also some hybrid approaches that take advantage of both techniques.

In terms of evaluating different similarity measures, we can either compare the correlated words to human judgments or select an application area of semantic similarity, and compare the results of different similarity measure according to the success rates in that application area [4]. In this project, we employ an extended version of Lesk algorithm to assign sense tags to Persian words.

4.2.2 WSD using the Lesk Algorithm

As it is described in section 2.4.1, the Lesk algorithm uses dictionary definitions (gloss) to disambiguate a polysemous word in a sentence context. The original algorithm counts the number of words that are shared between two glosses. The more overlapping the words, the more related the senses are.

To disambiguate a word, the gloss of each of its senses is compared to the glosses of every other word in a phrase. A word is assigned to the sense whose gloss shares the largest number of words in common with the glosses of the other words.

There are two hypotheses contributing to this approach [4]. The first one implies the words that appear together in a sentence can be disambiguated by assigning to them the senses that are most closely related to their neighbouring words. The idea behind this hypothesis is, the words that appear together in a sentence generally are related in some way, because to express some idea the human needs a set of words working together. The second hypothesis is that related senses can be identified by finding overlapping words in their definitions. Because, words that are related will often be defined using the same words, so their glosses will share common words.

The major limitation to this algorithm is that dictionary glosses are often quite brief, and may not include sufficient vocabulary to identify related senses. In order to overcome this limitation, we decided to apply an extended version of Lesk algorithm to our WSD task.

4.2.3 Extended Gloss Overlap

There is a more efficient version of Lesk algorithm [5] which extends the glosses of the concepts to include the glosses of other concepts to which they are related according to a given concept hierarchy. These related concepts are explicitly encoded in WordNet as relations, but can be found in any dictionary via synonyms, antonyms, or also-see references provided for a word sense. According to [5] this new measure reasonably correlates to human judgments. The next subsection gives a short overview of mentioned relations in WordNet.

4.2.3.1 Semantic Relations in WordNet

Synsets are connected to each other through explicit semantic relations that are defined in WordNet. These relations only connect word senses that are used in the same part of speech. Noun synsets are connected to each other through *hypernym*, *hyponym*, *meronym*, and *holonym* relations.

If a noun synset A is connected to another noun synset B through the *is–a–kind–of* relation then B is said to be a *hypernym* of synset A and A is a *hyponym* of B. For example the synset containing *car* is a hyponym of the synset containing *vehicle* and *vehicle* is a hypernym of *car*. If a noun synset A is connected to another noun synset B through the *is–a–part–of* relation then A is said to be a *meronym* of B and B a *holonym* of A. For example the synset containing *accelerator* is a meronym of *car* and *car* is a holonym of *accelerator*.

There are other types of relations between different part of speeches in WordNet, but we focused on these four types in our project. These relations are also applicable to Persian synsets in FarsNet. Unfortunately, these semantic relations are not available for us at the moment. To solve this issue, the mutual mappings between synsets in WordNet and FarsNet have been exploited. Our novel approach is described in more details in the next section.

Although WordNet provides *explicit* semantic relations between synsets, such as through the *is–a* or *has–part* links, it does not cover all possible relations between synsets [5]. For example, WordNet encodes no direct link between the synsets *car* and *tire*, although they are clearly related and interestingly their glosses have words in common. Hence those synsets can be considered to be related *implicitly*. Consequently, according to [5] given such a relation, we further conclude that synsets explicitly related to *car* are thereby also related to synsets explicitly related to *tire*. For example, we conclude that the synset *vehicle* (which is the hypernym synset of *car*) is related to the synset *hoop* (which is the hypernym synset of *tire*). Thus, the extended gloss overlap measure combines the advantages of gloss overlaps with the structure of a concept hierarchy to create an extended view of relatedness between synsets.

4.2.4 Applying Extended Lesk to Persian WSD

In order to compare the results of Direct and Cross Lingual approaches, the output from the cross lingual phase is fed as an input to the knowledge based phase. Each tagged word from phase 1 is considered as a **target word** to receive the second sense tag based on the extended Lesk algorithm.

For each target word, the sentence in which it appeared is also stored. Then, all the punctuations and stopwords are removed from each sentence and the neighbour words surrounding the target word are extracted. We call these words **content words**. In this project the window size is set to 5, so the target word will be located in the center and there will be _ at most _ two content words at each side. Some **morphological analysis** needs to be done for these content words, so the affixes will be removed which yields a better word overlap score. A simple morphological analyzer is developed for this project and is discussed in 4.2.4.2.

In general, for each word, either target word or content word, all relevant **Persian glosses are collected** using the mappings between FarsNet and WordNet synsets and the semantic relations in WordNet. The algorithm devised for retrieving these glosses is described in 4.2.4.3.

Once all Persian glosses are collected for the target word and the surrounding content words in each sentence, the overlap score will be calculated by comparing each sense of the target word to that of

each content word. The best sense will be the sense with the highest score. The **gloss overlap** algorithm is explained in 4.2.4.4.

4.2.4.1 Preprocessing

Prior to our WSD task some preprocessing needs to be performed for the input file. Once the input file is loaded, punctuations and stop words are removed from the tagged sentences. Two different lists of Farsi stop words were collected and modified to be adapted to our WSD task. After removing these stop words, the content words are extracted from the remaining words. Each of these content words needed to be converted to their base forms. The morphological analyzer utilized for this task is described in the next subsection.

4.2.4.2 Morphological Analyzer

We found one stemmer and one morphological lexicon [7] for Persian language: Perstem 0.9.7 and Perlex 0.0.1 respectively. Perstem is a free and open source stemmer and light morphological analyzer for Persian which is implemented in Perl. However, it worked very poorly for our texts. Since there was no other available Stemmer or Lemmatizer for Persian, a very simple morphological analyzer has been developed for this project. This morphological analyzer removes affixes including *Plural Sign*, *Possessive Pronoun* (which are attached to nouns and verb) and *Superlative Adjective Signs* from nouns.

In order to remove these affixes we need to have a lexicon which contains base forms of Persian words. We were not able to make use of Perlex. Therefore, a bilingual Persian to English dictionary which was available as a local database is employed to create this lexicon. In order to speed up the process of accessing this database, a Perl script is used to extract Persian words and write them out to a new file in XML format. Then this XML file is queried using XPath to check if an input Persian word exists in the lexicon. Similarly, a hash map data structure was implemented to speed up the process of accessing this lexicon. However, the improvement was not noticeable.

Stemming the words is the main bottleneck of our system which are used both for normalizing the content words (in the preprocessing phase) and calculating the score for glosses overlap. We will discuss about this issue later on.

There is a new package based on the work presented in [11] which we have been received after implementing this project. This system is called STeP-1 and can be considered as the first step in processing Persian texts as it performs a combination of tokenization, spell checking and morphological analysis. Testing this system as a part of our developed system will be one of the first items in our TODO list.

4.2.4.3 Collecting Persian Glosses

As it is mentioned before FarsNet includes semantic relations between synsets of the same part of speech in its first version. However, since it is not released now, we only had access to mappings between FarsNet synsets to their counterparts in WordNet. Hence, for extending the Farsi glosses the following approach has been followed:

For each word, the word ID (WID) is sought from the FarsNet's lexicon file. Then this ID is passed to the synset file in order to retrieve all Farsi synsets containing this word from FarsNet and their equivalent (mapped) synsets in WordNet. Next, for each mapped English synset offset in WordNet, the database files for different part of speeches are sought to retrieve the synset information. Synsets information is

available in different lexicographer files in WordNet database. In this project we only work with noun and adjective files. In these files, each line corresponds to a synset. The general synset syntax is:

Synset ID, Words, Pointers, Gloss

After finding each synset using Synset ID (offset), we parse it and extract the required information. For each synset we store ID, words, pointers and gloss. Pointers are used to represent the relations between the words in one synset and another. Semantic pointers represent relations between word meanings, and therefore pertain to all of the words in the source and target synsets. Lexical pointers represent relations between word forms, and pertain only to specific words in the source and target synsets.
We only need semantic pointers of type hypernym, hyponym, meronym and holonym. The rest will just be ignored. Once these pointers are retrieved we have the offset of all related synset to the current English synset. So now we can go back to FarsNet and retrieve the Farsi mapped synsets for these English ones.
Finally, for each sense of the current word, we retrieve all related synsets from FarsNet. Now we need to collect all Persian glosses for each sense. To do so, all glosses of the same type (for example all glosses from the synsets which are in a hypernym relation with the current synset) are concatenated. As a result, each synset has at most five glosses: gloss, hypernym, hyponym, meronym and holonym.

4.2.4.4 Gloss Overlap Algorithm

Having all Persian glosses collected for each sense of target word and also each sense of each content word, we need to form two sets of glosses for each pair of senses to be overlapped. Following [5], we form a non–empty set of *pairs* of relations from the set of relations as:

$$RelPairs = \{(R1, R2) | R1, R2 \in \{gloss, hype, hypo, mero, holo\}\}$$

Please note that each of the relation types of hype, hypo, mero and holo covers all glosses of that type. For example, hype means all glosses of the synsets which are related to the source synset through a hypernym relation which are concatenated as a single gloss. The reason [5] for this concatenation is that we do not wish to differentiate between the different synsets that are all related to the input synset through a particular relation, but instead are only interested in all their definitional glosses.

In order to make sure the relatedness measure is reflexive, for every relation pair $(R1, R2), R1 \neq R2$ we also add $(R2, R1)$. Then the relatedness between two synsets A and B (two senses of two words) are defined as follows:

$$\begin{aligned} relatedness(A, B) \\ &= score\big(gloss(A), gloss(B)\big) + score\big(gloss(A), hype(B)\big) \\ &+ score\big(gloss(B), hype(A)\big) + score\big(hype(A), hypo(B)\big) \\ &+ score\big(hypo(A), hype(B)\big) + score\big(hype(A), hype(B)\big) + \dots \end{aligned}$$

Observe that due to the measure's reflexivity feature, $relatedness(A, B) = relatedness(B, A)$.

Now we need to apply the concept of *semantic relatedness* to our WSD task. To this end, we assign to each possible sense k of the target word a $SenseScore_k$ computed by adding together the relatedness

scores obtained by comparing the sense of the target word in question with every sense of every content word in the window of context. The *SenseScore* for sense $s_{0,k}$ is computed as:

$$SenseScore_k = \sum_{i=-n}^{n} \sum_{j=1}^{|w_i|} relatedness(s_{0,k}, s_{i,j}), i \neq 0$$

The sense with the highest *SenseScore* is considered to be the most appropriate sense for the target word. If there are on average a senses per word and the window of context is N words long, there are $a^2 \times (N-1)$ pairs of sets of synsets to be compared, which increases linearly with N [5].

If two senses yield the same score, we store both of them, one as the best sense and the second one as an alternative sense. Currently, if no sense receives a positive score we don't assign any sense. However, a better approach will be assigning either a random sense or the most frequent sense. Since FarsNet does not have the most frequent sense implemented yet, we will use this feature once it is available.

4.3 Comparison: Knowledge based vs. Cross Lingual

Once we decided to apply the Extended Lesk algorithm to Persian WSD we were to disambiguate the same set of tagged words from Cross Lingual approach, so we can compare the performance of these two approaches for our WSD task. However, while implementing the new approach, we had to make some simplifying assumptions and add some constraints to this algorithm. Therefore, the outputs of these two approaches are not completely comparable.

First of all, the cross lingual approach performs an All-word disambiguation, while the knowledge based one disambiguates only target words. However, these target words are indeed the words which are tagged by the first method and their tags thus can be compared to those of assigned by the second method.

Second, since the running time of this system was high, we had to trade some coverage to gain some speed. Therefore, we decided to tag each target word based on only one of the sentences in which it appeared. So, we are able to compare the tags assigned by both methods only for one occurrence of each word.

While in terms of the overall accuracy of these two approaches we cannot prefer one over the other, we can compare them based on simplicity, use of resources and performance. The main advantage of the cross lingual method is it is fast and simple. SenseRelate is implemented in Perl and assigns sense tags to English words of a page in seconds. The transferring the senses to Persian words is also implemented in Perl and is very fast too. Besides, the whole algorithm is very simple. What is more, SenseRelate can be replaced with any other English sense disambiguator. So, we can improve both accuracy and coverage by using a more efficient English sense tagger, while the Persian side remains intact.

On the other hand, this approach assigns the same tag to all occurrences of a word which is reasonable at all. Besides, it is only applicable to some parallel corpus like Wikipedia. So, if there is no English text with the same context is available for a Persian corpus, this method cannot be applied. What is more, when the bilingual texts are not the direct translation of one another the system coverage will be

limited to *common* words in both English and Persian texts. So, it mainly works well for content words and not for all the words appearing in the Persian texts.

The main downside of the implemented Lesk based method is its running time. For each target word we have 4 content words and for each word we have a senses on average. As we extend the glosses and consider all the glosses in a relation with the current one, we can estimate the number of glosses for each sense of a word as the $number\ of\ associated\ relations$. Since we concatenate all the glosses of the same type as one single gloss and we focus only on five different types of relations $\{gloss, hype, hypo, mero, holo\}$ we have at most 5 glosses for each sense of the word. However, some of them can be really long. Then for each sense of each content word we need to set the relation pairs with each sense of the target word and overlap them. So in other words we have $5 \times 5 = 25$ pairs of glosses for $a \times a$ different combinations of senses of the target word with each content word. The total number of gloss overlaps for assigning each sense tag to a target word thus can be calculated as:

$$count_{content\ words} \times count_{Senses_{content\ word}} \times count_{Senses_{target\ word}} \times count_{relation\ pairs} = 4 \times a \times a \times 25 = 100\ a^2$$

For each of these gloss overlaps we need our morphological analyzer to extract the morph of each word. This will yield a better score in terms of number of common words between two glosses. This is indeed the main bottleneck of the implemented system which has slowed down the process a lot. We tried four different implementations for accessing the lexicon in order to speed up the gloss overlap procedure. However, overlapping the glosses is still a challenge for our project. These different implementations are included in our commented codes.

The other problem for applying this algorithm to Persian WSD is that we need to perform NLP approaches for Farsi texts which has always been a challenge for this language. There were no available Tokenizer, Stemmer and POS tagger for Farsi at the moment. So they are either implemented from scratch (tokenizer and morphological analyzer) or ignored (pos tagger).

5 Challenges for Persian

Any system which deals with Persian needs to overcome many different challenges. One of the primary obstacles is lack of annotated data and knowledge resources. Once FarsNet as a semantic lexicon for Persian is released it can definitely contribute to many projects for this language.

Lacking sense tagged corpora for Farsi texts, we had to evaluate our system's results manually which is a tedious and error prone task. Besides, we could only use plain un-tagged texts as inputs to our system which affects the performance in terms of accuracy, recall and running time. No available POS tagger for Farsi, made us ignore the part of speech of the target word we want to disambiguate. When the part of speech is not known, we need to use relations and synsets associated with all the possible parts of speech. However, if we know the part of speech of the word, or if the word is only used in a subset of the possible parts of speech, then the number of relation pairs considered is less. Since we did not have a POS tagger, for each word or its English translation can be more than one possible parts of speeches

and we had to consider them all which dramatically increases the number of associated glosses and different number of senses for that word.

6 Evaluation

The assessment of word sense disambiguation systems is usually performed in terms of evaluation measures borrowed from the field of information retrieval [1].

Precision P determines how good the answers given by the system being assessed are. *Recall R* is defined as the number of correct answers given by the automatic system over the total number of answers to be given:

$$P = \frac{\#\text{ correct answers provided}}{\#\text{ answers provided}}$$

$$R = \frac{\#\text{ correct answers provided}}{\#\text{ total answers to provide}}$$

6.1 Cross-Lingual Approach

The ultimate results of this method have been evaluated on 7 Wikipedia pages which were available in both languages and 366 distinct Persian words were received sense tags. Evaluation results indicate an accuracy of 84% for these pages. Table 1 summarizes these results.

	number	Percentage
The best sense assigned	308	84%
Almost Accurate	31	8%
Wrong sense assigned	27	8%
Total number/percentage of disambiguated words	366	100%

Table 1. Evaluation Results for Cross-lingual approach

Studying the output results, it turned out that the domain-specific words which usually occur frequently in both English and Persian texts, are highly probable to receive the correct sense tag.

Since the number of Persian Wikipedia pages is still limited, these pages usually have relatively shorter texts, and finally, the Persian texts are not direct translation of the English ones, this system suffers from low Recall. However, as Wikipedia covers more and more Persian pages every day, soon we will be able to overcome this bottleneck.

In this project we didn't evaluate our system in terms of Recall and it is one of the primary tasks in our TODO list.

WSD systems are commonly evaluated by comparing their results to some *Baselines.* A baseline is a standard method to which the performance of different approaches is compared. This system can be further evaluated by comparing its output to the results of assigning either random senses or the first sense to words. Since the senses in FarsNet are not sorted based on their frequency of usage (as opposed to WordNet), we were not able to use this baseline. Instead, we decided to evaluate the results by comparing the assigned Persian sense tags to the first sense appearing in FarsNet for each word with

the same part of speech. Although the first sense in FarsNet does not correspond to the most frequent sense according to Persian texts, it can act as a baseline for our WSD approach.
Assigning the first sense to all tagged Persian words, the performance decreased significantly in terms of accuracy. These new results are summarized in Table 2.

	Percentage
The best sense assigned	56%
Almost Accurate	9%
Wrong sense assigned	35%

Table 2. Evaluation Results for Cross-lingual approach – Applying baseline

According to the results indicated in Table 2, applying our novel approach will result in a 28% improvement in accuracy in comparison with this selected baseline. However, assigning the most frequent sense to Persian words would be a more realistic baseline which yields a better estimation for our system's performance. Thus by the time the frequency of usage is provided for FarsNet senses, we will be able to take advantage of this more standard baseline.

6.2 Direct Knowledge-based approach

As it is mentioned before, we tag the same tagged word by the first approach again with the knowledge based method in order to compare their results.

According to the limited time we had for this project we evaluated the results of the new approach for 155 target words. Overall, 56% of the tags received a different tag using the second approach. Table 3 indicates the evaluation results for the different tags.

	Cross-Lingual	Knowledge-based
The best sense assigned	40%	18.6%
Almost Accurate	23%	38%
Wrong sense assigned	37%	43%

Table 3. Evaluation Results for comparing two approaches

According to the partially evaluated results the Cross-lingual approach gained better results in terms of more accurate sense assigned. However, this does not imply that the Cross-lingual approach performs better than the Extended-Lesk based one. We still need to evaluate the results for more tags and there are many parts in the Knowledge based method which can be improved to a good extent.

7 Conclusion and Future Work

One of the primary challenges in Machine Translation is the *Word Sense Disambiguation* (WSD) problem. A *word sense* is a commonly accepted meaning of a word. While the words are changing through translation process, we need to maintain the sense.

There are many available resources and Text Understanding tools for English language, whereas when it comes to Persian language, there is neither any semantically tagged corpus to aid Machine Learning approaches for Persian texts, nor any parallel corpora. In this project we aimed to overcome this problem by taking advantage of available English sense disambiguators, some special features – such as availability of articles in both languages – in Wikipedia and the newly developed lexical ontology, FarsNet, in order to address the WSD issue for Persian. The evaluation results of the Cross-lingual approach show a 28% improvement in accuracy in comparison with the first-sense baseline being used and it performed better than the knowledge based approach which is directly applied to Persian sentences. However, one of the main reasons for this is that lack of NLP tools and comprehensive knowledge resources for Persian introduces many challenges for the systems dealing with this language.
We encountered many challenges throughout this project. The number of Farsi pages in Wikipedia is still very limited and they usually contain short texts. Moreover, the English disambiguators are not 100% accurate, so we needed to manually correct the errors from the English WSD phase in order to prevent accumulative errors. Similarly, since FarsNet is not released yet, the data contains errors both in its synsets and the mappings to WordNet synsets. What is more, this lexical ontology doesn't have the full coverage neither for the synsets nor for the mappings. As a consequence, the performance of our system is affected by all these issues.

This project in the first step has aimed at examining a novel idea for cross-lingual WSD in terms of plausibility, feasibility and performance. There are different ways to improve our system. First of all we can try different English disambiguators to find the most accurate system, which in turn increases the ultimate accuracy. As it mentioned before, supervised approaches usually lead to more accurate results. Therefore, finding a suitable dataset which is specialized for sense disambiguation purpose and trying different classification methods in order to assign different occurrences of words (instances) to appropriate senses (classes) can increase the English WSD accuracy the most.

Second, we can improve the results of the knowledge based approach by using STeP-1 for both tokenizing and stemming the words. Third, once the semantic relations are available for us to use, we can simplify this system by using them directly instead of using the WordNet relations and then mappings them back to FarsNet. This will definitely improve the system's performance and decreases its running time.

Improving the accuracy of our system, we can apply some bootstrapping methods to use our system's output (sense-tagged Persian texts) and some un-tagged texts from the web, in order to gradually grow the sense-labeled texts which can be utilized as a training set for other Persian sense disambiguator systems to benefit from for Classifying new texts.

Finally we need to evaluate our system further, with more Wikipedia pages and calculate the Recall criterion for them. Due to the limited time we had for this project, and the very time-consuming process of manually evaluating the results, I had to confine the number of input pages to 7. Therefore, evaluating the results by more human evaluators will be of great benefits. As Farsi pages in Wikipedia are growing and FarsNet is being finalized we can have a better estimation of our system performance.

8 References

[1] R. Navigli, *Word Sense Disambiguation: A Survey*, ACM Computing Surveys, 2009.

[2] M. Shamsfard, *Developing FarsNet: A Lexical Ontology for Persian*, 4th Global WordNet Conference (GWC'08), 2008.

[3] V. Kolhatkar, *An Extended Analysis of a Method of All Words Sense Disambiguation,* - Master of Science Thesis, Department of Computer Science, University of Minnesota, Duluth, August, 2009.

[4] S. Torres, A. Gelbukh, *Comparing Similarity Measures for Original WSD Lesk Algorithm*, Advances in Computer Science and Applications, Research in Computing Science 43*, 2009*

[5] S. Banerjee , T. Pedersen, *Extended Gloss Overlaps as a Measure of Semantic Relatedness*, In Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, 2003

[6] S. Banerjee, T. Pedersen, *An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet*, Published in Springer-Verlag Berlin Heidelberg, 2002

[7] B. Sagot, G. Walther, *A Morphological Lexicon for the Persian Language*, In Proceeding of LREC, 2010

[8] T. Pedersen, V. Kolhatkar, *WordNet::SenseRelate::AllWords -A Broad CoverageWord Sense Tagger that Maximizes Semantic Relatedness*, Proceedings of NAACL HLT 2009, Boulder, Colorado, June 2009

[9] M. Hearst, *Noun homograph disambiguation using local context in large text corpora,* In Proceedings of the 7th Annual Conference of the UW Centre for the New OED and Text Research: Using Corpora (Oxford, U.K). 1–19, 1991

[10] D. Yarowsky, *Unsupervised word sense disambiguation rivaling supervised methods,* In Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (Cambridge, MA). 189–196, 1995

[11] M. Shamsfard, S. Kiani, Y. Shahedi, *STeP-1: Standard Text Preparation for Persian Language*, In Proceedings of Machine Translation Summit XII, Ottawa, Ontario, Canada, 2009

8.1 Useful Links

Perstem project: http://sourceforge.net/projects/perstem/

PerLex project: http://alexina.gforge.inria.fr

FarsNet: http://nlp2.sbu.ac.ir/projects/farsnet

Some NLP tools to download for many different languages:
http://members.unine.ch/jacques.savoy/clef/index.html

9 Appendix

9.1 User Manuals

9.1.1 EWSD

This manual describes Perl-scripts toolkit for cross-lingual English-Persian Wikipedia word sense disambiguation.

Contents

When the distribution is unpacked, several subdirectories are created:

/helpers
This directory contains test versions of offset and target extracters that were intended to be used with target word sense disambiguation.
/farsnet
This directory contains FarsNet XML database files that are used in disambiguation.
/input
This directory contains plain untagged english and persian text files to be disambiguated.
/output
This directory contains several types of output files produced during the course of disambiguation:

1. Files produced by disambiguate.sh script:
   - topic_name.glosses.txt
     Contains WordNet definitions of disambiguated words.
   - topic_name.neat.txt
     Contains disambiguated text in a format that is more human-friendly.
   - topic_name.out.txt
     Contains sense-tagged text.
2. Files produced by preprocess.sh script:
   - topic_name.split.txt
     Contains text that has been preprocessed to meet requirements of SenseRelate.
3. Files produced by extractOffsetBetter.pl script:
   - topic_name.neat.txt.offset
     Contains sense-tagged text along with corresponding WordNet sense offsets.
   - topic_name.neat.txt.validoffset
     Same as above, but words in text are lemmatized.
4. Files produced by extractPersian.pl
   - topic_name.neat.txt.offset.persian
     Contains, in addition to english senses, corresponding persian senses and definitions that have been fetched from FarsNet.
5. Files produces by parsePersian.pl script
   - topic_name.neat.txt.offset.persian.final
     For each disambiguated persian word contains the following:
     - Persian word
     - It's synset ID in FarsNet
     - It's English Word equivalent's offset in WordNet
     - Assigned gloss
     - All sentences this persian word appears at. (For evaluation purposes)

/test

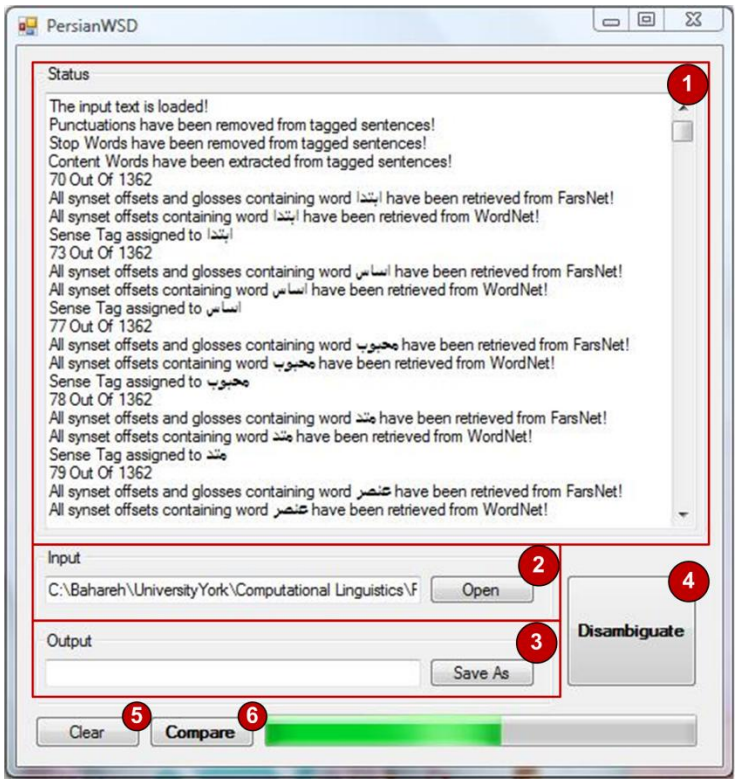
This directory contains some test Perl scripts.

Example Usage

1.) From WikiPedia, extract English and Persian pages for a desired topic. Save corresponding untagged text in topic_name_en.txt and topic_name_fa.txt files.
2.) Run preprocess script for English and Persian pages:
```
./preprocess.sh topic_name_en.txt
./preprocess.sh topic_name_fa.txt
```
3.) Run disambiguate script for English page (this may take a while):
```
./disambiguate.sh topic_name_en.txt
```
4.) Run offset extractor script for English "neat" page:
```
./extractOffsetBetter.pl topic_name_en.neat.txt
```
5.) Run persian extractor script for English offset page (this may take a while):
```
./extractPersian.pl topic_name_en.neat.txt.offset
```
6.) Run final persian disambiguation script:
```
./parsePersian.pl topic_name_en.neat.txt.offset.persian
topic_name_fa.split.txt
```
7.) Alternatively, use script that will process all .persian files at-once:
```
./parsePersianAll.sh
```

9.1.2 PersianWSD

1. The status box shows the current status of the running application.
2. The user can specify the file to load the input text from.
3. The user can specify the location to save the output.
4. This button launches the application.
5. Restores all fields to their default values.
6. Calculates the statistics in order to compare the results of two approaches.



9.2 Code Listing

9.2.1 EWSD

9.2.1.1 preprocess.sh

```
#!/bin/bash

# Strip path and extension from argument
filename=${1##*/}
filename=${filename%.*}

#Split input file into sentences, one sentence per line
./sentence_split.pl --context $1 > output/$filename.split.txt

#Do some preprocessing with Perl
# Remove leading list stars "* list item"
perl -pi -e "s/^\*\s+//" output/$filename.split.txt

# Remove sandwiched list stars "list item  * list item"
perl -pi -e "s/\s+\*\s+/ /" output/$filename.split.txt

# Remove numbers "[number]"
perl -pi -e "s/\[\d+\]//" output/$filename.split.txt

# Remove "[edit]"
perl -pi -e "s/\[edit\]//" output/$filename.split.txt

# Remove blank lines
perl -pi -e "s/^\n//" output/$filename.split.txt

# Remove 1 word sentences
perl -pi -e "s/^\w*\n$//" output/$filename.split.txt

# Remove extra spaces
perl -pi -e "s/\s+/ /" output/$filename.split.txt

# Remove leading spaces
perl -pi -e "s/^\s+//" output/$filename.split.txt

# Remove all non-alphanumeric
#perl -pi -e "s/\W//" output/$filename.split.txt

```

9.2.1.2 disambiguate.sh

```
#!/bin/bash

# Strip path and extention from argument
filename=${1##*/}
filename=${filename%.*}

# Disambiguate without glosses
./wsd.pl --context output/$filename.split.txt --format raw
    --stoplist default-stoplist-raw.txt --window 5
    --outfile output/$filename.neat.txt > output/$filename.out.txt &

# Disambiguate with glosses
./wsd.pl --context output/$filename.split.txt --format raw
    --stoplist default-stoplist-raw.txt
    --window 5 --glosses > output/$filename.glosses.txt &

```

9.2.1.3 extractOffsetBetter.pl

```
#!/usr/bin/perl -w
use strict;
use warnings;
use WordNet::QueryData;

my $contextFilePath;
my $contextLine;
my $wn = WordNet::QueryData->new( noload => 1);

$contextFilePath = $ARGV[0];
print "$contextFilePath\n";

open CONTEXT, $contextFilePath or die "Could not read from $contextFilePath, program halting.";
open OUT, ">$contextFilePath.offset" or die "Could not read from $contextFilePath, program halting.";
open VALIDOUT, ">$contextFilePath.validoffset" or die "Could not read from $contextFilePath, program halting.";

print "Loaded files \n";

# Read context file line-by-line
while ($contextLine = <CONTEXT>)
{
    #print $contextLine;

    # Extracting the following:
    # base_form#pos#sense_assigned#offset_in_WordNet
    # If the word has been disambiguated
    if ($contextLine =~ /(\w+)\s+(\w+)\s\s(\w)\s\s(\d)/ )
    {
        my $offset = $wn->offset("$2#$3#$4");
        print VALIDOUT "$2#$3#$4#$offset ";
        print OUT "$1#$3#$4#$offset ";
    }
    # If the word has not been disambiguated
    elsif ($contextLine =~ /(\w+)\s\s(\w+)/)
    {
        print VALIDOUT "$1#$2 ";
        print OUT "$1#$2 ";
    }
    # End of sentence
    elsif ($contextLine =~ /Results/)
    {
        print VALIDOUT "\n";
        print OUT "\n";
    }

}

close(CONTEXT);
close(OUT);
close(VALIDOUT);

print "All done! \n";
```

9.2.1.4 extractPersian.pl

```
#!/usr/bin/perl -w
use strict;
use warnings;
use WordNet::QueryData;
use XML::Twig;

binmode STDOUT, ":utf8";

# FarsNet data files
my $synsetXML = 'farsnet/synsets0.xml';
my $wordXML = 'farsnet/words0.xml';

# Read file path of file to process as an argument from a user
my $contextFilePath = $ARGV[0];

# Misc variables
my $word;
my @words;
my $contextLine;
my $offset;
my $isFound;

# Load FarsNet data files into data structures
my $twigSynset = XML::Twig->new();
my $twigWord = XML::Twig->new();
$twigSynset->parsefile($synsetXML);
$twigWord->parsefile($wordXML);

# Open files to read/write
open CONTEXT, $contextFilePath or die "Could not read from $contextFilePath, program halting.";
open OUT, ">$contextFilePath.persian" or die "Could not read from $contextFilePath, program halting.";
binmode OUT, ":utf8";

# Get root of XMLs
my $rootSynset = $twigSynset->root;
my $rootWord = $twigWord->root;

while ($contextLine = <CONTEXT>)
{
    #print $contextLine;

    # Split line into words
    # In this case "word" is the input from last script in form:
    # base_form#pos#sense_assigned#offset_in_WordNet
    # if the word has been disambiguated
    @words = split / /, $contextLine;

    # Process word-by-word
    foreach $word (@words)
    {
        #print $word;

        # If the word has been disambiguated
        if($word =~ /(\w+#\w#\d)#(\d+)/ )
        {
            $isFound = 'false';
            my $rest = $1;
            $offset = $2;
            if ($offset =~ /^0+([1-9]\d*)/)
            {
```

```
61                  $offset = $1;
62              }
63
64              # From FarsNet, extract the gloss and all synonyms based on WordNet offset
65              foreach my $synset ($rootSynset->children('SYNSET'))
66              {
67                  my $synsetID = $synset->first_child_text('ID');
68                  my $glossPersian = $synset->first_child_text('GLOSS');
69                  my $wnSynId = $synset->first_child('MappedWNsynId');
70                  my $offsetXML = $wnSynId->first_child_text('OFFSET');
71
72                  # If the synset and gloss has been found
73                  if ($offset eq $offsetXML)
74                  {
75                      $isFound = 'true';
76                      print "$rest#$offset#$synsetID#";
77                      print OUT "$rest#$offset#$synsetID#";
78                      my $senses = $synset->first_child('SENSES');
79                      foreach my $sense ($senses->children('SENSE'))
80                      {
81                          my $wid = $sense->first_child_text('WID');
82
83                          # Output all persian synonyms
84                          foreach my $word ($rootWord->children('WORD'))
85                          {
86                              if($wid eq $word->first_child_text('WID'))
87                              {
88                                  my $wordValue = $word->first_child_text('WORDVALUE');
89                                  print "$wordValue#";
90                                  print OUT "$wordValue#";
91                              }
92                          }
93
94                      }
95
96                      # Output persian gloss
97                      print OUT "#";
98                      print OUT "\"$glossPersian\" ";
99                      print "#";
100                     print "\"$glossPersian\" ";
101                 }
102                 else
103                 {
104                     # do nothing
105                 }
106             }
107
108             # If the synset and gloss has not been found
109             if ($isFound eq 'false')
110             {
111                 print OUT "$rest#$offset ";
112                 print "$rest#$offset ";
113             }
114         }
115
116         # If the word has not been disambiguated
117         else
118         {
119             print "$word ";
120             print OUT "$word ";
121         }
122
123     }
```

```
    print "\n";
}

close(CONTEXT);
close(OUT);
```

9.2.1.5 parsePersian.pl

```
#!/usr/bin/perl -w

use strict;
use warnings;

my $contextFilePath = $ARGV[0];
my $persianFilePath = $ARGV[1];
my $contextLine;
my @defs;
my $defn;
my $word;
my @words;
my %usedWords = ();

open CONTEXT, $contextFilePath or die "Could not read from $contextFilePath, program halting.";
open OUT, ">$contextFilePath.final" or die "Could not read from $contextFilePath.final, program halting.";

binmode OUT, ":utf8";
binmode CONTEXT, ":utf8";
binmode STDOUT, ":utf8";

while ($contextLine = <CONTEXT>)
{
    @defs = split /" /, $contextLine;
    foreach $defn (@defs)
    {
        if ($defn =~ /\w+#\w+#\d+#(\d+)#(\d+)#(.+)##"(.+)/ )
        {
            @words = split /#/, $3;
            foreach $word (@words)
            {
                if(exists $usedWords{$word})
                {
                    # Do nothing
                }
                else
                {
                    $usedWords{$word} = 0;

                    print OUT "--- Word ---\n";
                    print OUT "$word\n";
                    print OUT "Synset ID: $2\n";
                    print OUT "Offset: $1\n";
                    print OUT "--- Gloss ---\n";
                    print OUT "$4\n";
                    print OUT &findSentences($persianFilePath, $word);
                }
            }
            print OUT "\n";
        }
    }
}

sub findSentences
{
```

```
    my ($filePath, $word) = @_;
    my $returnString;
    my $persianLine;

    open PERSIAN, $filePath or die "Could not read from $filePath, program halting.";
    binmode PERSIAN, ":utf8";

    while ($persianLine = <PERSIAN>)
    {
        if($persianLine =~ /$word/)
        {
            $returnString .= "--- Sentence ---\n";
            $returnString .= "$persianLine\n";
        }
    }

    close(PERSIAN);

    return $returnString;
}

close(CONTEXT);
close(OUT);
```

9.2.1.6 parsePersianAll.sh

```
#!/bin/bash

for file in `ls output/*.persian` ; do
    ./parsePersian.pl $file ${file%_*}_fa.split.txt
done
```

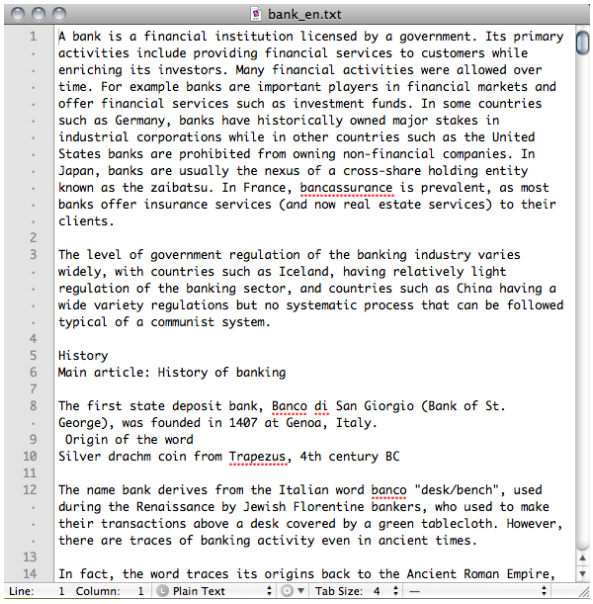
9.2.2 PersianWSD

PersianWSD code listing goes here…

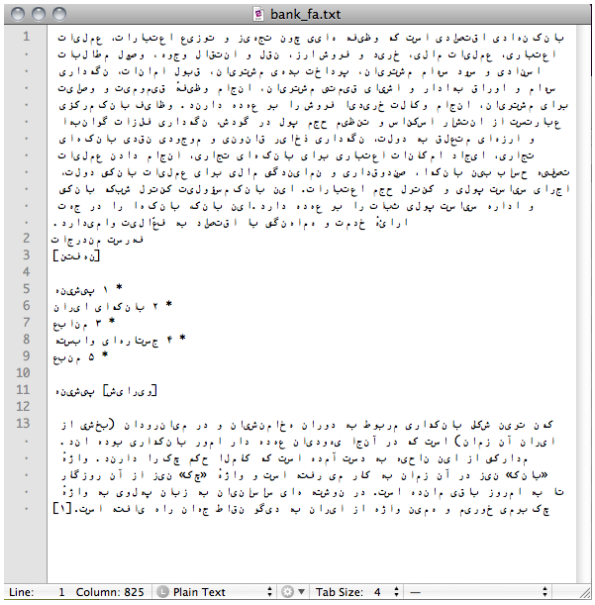
9.3 Sample Input/Output

9.3.1 EWSD

1.) Sample English Wiki Page



2.) Sample Farsi Wiki Page



3.) Sample Output

For each disambiguated word the following is displayed:

- Persian word
- It's synset ID in FarsNet
- It's English Word equivalent's offset in WordNet
- Assigned gloss
- All sentences this persian word appears in. (For evaluation purposes)

9.3.2 PersianWSD

Sample input: The input is taken from EWSD output.

Sample output: