FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Petr Kašpárek

Cross-lingual transfer for the annotation of the SynSemClass ontology

Institute of Formal and Applied Linguistics

Supervisor of the bachelor thesis:	prof. RNDr. Jan Hajič, Dr.
Consultant of the bachelor thesis:	RNDr. Jana Straková, Ph.D.
Study programme:	Computer Science

Prague 2024

Dedication

I would like to thank my supervisor, prof. RNDr. Jan Hajič, Dr., for suggesting the thesis topic, giving me the opportunity to work on this project and also his help and advice throughout my work on the thesis.

I also would like to thank the consultant of my thesis, RNDr. Jana Straková, Ph.D., for her dedicated help and advice, both on the technical implementation as well as on academic writing. I greatly appreciate the time she spent on giving me advice, detailed explanations of technical problems, as well as the time spent proof-reading the thesis.

I would also like to thank my family.

Title: Cross-lingual transfer for the annotation of the SynSemClass ontology

Author: Petr Kašpárek

Institute: Institute of Formal and Applied Linguistics

Supervisor: prof. RNDr. Jan Hajič, Dr., Institute of Formal and Applied Linguistics

Consultant: RNDr. Jana Straková, Ph.D., Institute of Formal and Applied Linguistics

Abstract: This work compares two approaches to automatic preannotation of semantic class to verbs in a sentence for the purpose of adding a new language to the SynSemClass ontology. Both approaches rely on a multilingual deep learning classification model fine-tuned on already annotated English, Czech and German data of the ontology. The first, more classical, approach is annotation projection. It uses a parallel corpus and the aforementioned model to make predictions on a source language already present in the ontology and projects the predictions onto the target language using automated word alignment. The second approach, zero-shot cross-lingual transfer, assumes that the multilingual properties of the underlying model are sufficient and that we can make reasonable predictions directly on the target language, even though the model was never trained for that specific task on the specific target language. For the purpose of evaluation, we manually build and annotate a small Korean language dataset to test the performance on a language significantly different from English, Czech and German. We conclude that the zero-shot approach performs notably better than the alignment approach ($p < 0.005$) obtaining 0.54 both in recall and precision, compared to 0.37 and 0.41 in recall and precision respectively of the alignment approach. We perform an analysis of the errors and find that the extra steps of annotation projection introduce cascading errors and that loose translation poses a problem in itself.

Keywords: annotation projection, zero-shot cross-lingual transfer, ontology, multilingual natural language processing, lexical semantics

Název práce: Mezijazykový transfer pro anotaci SynSemClass ontologie

Autor: Petr Kašpárek

Institut: Ústav formální a aplikované lingvistiky

Vedoucí bakalářské práce: prof. RNDr. Jan Hajič, Dr., Ústav formální a aplikované lingvistiky

Konzultantka: RNDr. Jana Straková, Ph.D., Ústav formální a aplikované lingvistiky

Abstrakt: Tato práce porovnává dva přístupy k automatické předanotaci sémantických tříd sloves ve větách za účelem přidání nového jazyka do ontologie SynSemClass. Oba přístupy vycházejí z vícejazyčného deep learning klasifikačního modelu, který byl fine-tunovaný na již anotovaných anglických, českých a německých datech z ontologie. První, více tradiční, přístup je annotation projection. Používá paralelní korpus a výše zmíněný model k vytvoření predikcí na zdrojovém jazyce, který je již obsažen v ontologii, a tyto predikce projektuje na cílový jazyk pomocí automatického word alignmentu. Druhý přístup, zero-shot cross-lingual transfer, předpokládá, že vícejazykové schopnosti deep learning modelu jsou dostatečné a že můžeme vytvořit kvalitní predikce přímo na cílovém jazyce, i když model nebyl nikdy trénován pro danou úlohu na daném cílovém jazyce. Pro účely vyhodnocení ručně vytváříme a anotujeme malý korejský dataset za účelem otestování výsledků na jazyce, který se významně liší od angličtiny, češtiny a němčiny. Dospíváme k závěru, že zero-shot transfer vykazuje výrazně lepší výkon než annotation projection ($p < 0{,}005$), s hodnotami recall a precision 0,54, ve srovnání s 0,37 recall a 0,41 precision u annotation projection. Také provádíme analýzu chyb a zjišťujeme, že dodatečné kroky annotation projection zavádějí kaskádovité chyby a že volný překlad sám o sobě představuje problém.

Klíčová slova: annotation projection, zero-shot cross-lingual transfer, ontologie, vícejazyčné zpracování přirozeného jazyka, lexikální sémantika

Contents

Introduction

Semantic ontologies are important linguistic resources that capture our knowledge of the world. Such lexical resources provide a solid foundation for theoretical linguistic research and become building blocks for other natural language processing research. We work with the multilingual ontology SynSemClass (Urešová et al., 2020) that catalogues semantic classes of word senses. Currently, it only contains verbs.

Manually annotating such ontology is costly. Previously, the SynSemClass project utilized existing lexical resources to build up and extend the ontology. However, rich lexical resources are rare and not harmonized into the same format and a common annotation guideline, thus their usage cannot be easily streamlined. We want to be able to easily extend the ontology for a new language without these resources. In order to do this, we implement a toolchain that creates automatic annotation suggestions. These suggestions will be then given to human annotators who will review them and integrate the correct suggestions into the ontology and discard the erroneous suggestions.

To this end, we use existing tools and evaluate two methods of using these tools. Both of these methods rely on a deep learning classification model for SynSemClass (Straková et al., 2023). The first method, annotation projection, utilizes a sentence-aligned parallel corpus. The method consists of generating annotations on a source language natively supported by the classification model and projecting those automatic annotations onto the target language using automatic word alignment. The second method, zero-shot cross-lingual transfer, relies on the multilingual pretraining of the classification model and generates predictions directly on the target text.

To evaluate the performance of these two methods, we chose Korean as our target language. Korean is an agglutinative head-final East Asian language. This sets it apart from the European head-initial languages that are currently part of the SynSemClass ontology, namely English, Czech, German and Spanish. We build a small Korean–English parallel corpus and manually annotate it for semantic classes and word alignment. We then measure the performance for both methods and verify that the difference is statistically significant.

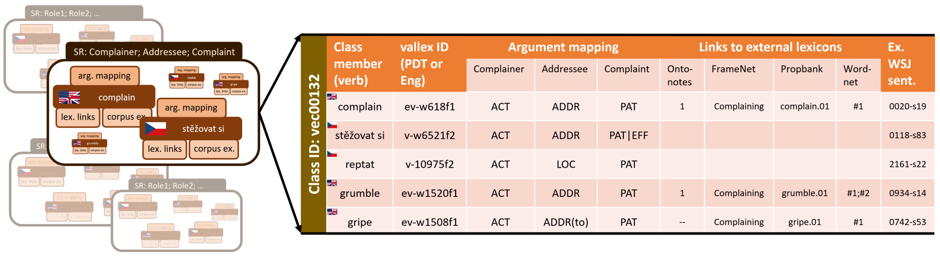
After discovering that zero-shot cross-lingual transfer performs significantly better, we dive deeper into the reasons why. We find that the loose non-verbatim translation poses a fundamental limit to annotation projection. The text is often rephrased such that the verbs we want to project often change meaning or are eliminated entirely. Furthermore, we show that errors in word alignment contribute to the overall error. We conclude the classification model performs roughly the same on both Korean and English under ideal conditions, but the extra steps of annotation projection introduce cascading errors.

The last step is creating the final annotation suggestions. Since the classification model creates predictions on single mentions of verbs, our goal is to aggregate the individual predictions into a single suggestion for each class-lemma pair. In section 3.2, we investigate what the most efficient aggregation strategy is.

1. Background

1.1 SynSemClass

The SynSemClass lexicon (Urešová et al., 2020) — formerly CzEngClass — is a multilingual event-type ontology currently under development. An event-type ontology is a hierarchy of classes that denote events and states. Each class is populated by class members — words. Unrelated to our work, the SynSemClass project also seeks to link the classes and its members to a plethora of existing lexical resources, such as the EngVallex (Cinková et al., 2021), CzeEngVallex (Urešová et al., 2016), FrameNet (Ruppenhofer et al., 2016), PropBank (Kingsbury and Palmer, 2002) and many others. The lexicon currently consists of words from English, Czech, German and Spanish. See figure 1.1 for the scheme of the lexicon and figure 1.2 for the tools used to view the ontology data. The goal of this bachelor thesis is to explore machine learning methods as preprocessing steps to automatically suggest annotations for extending the ontology into a new language.

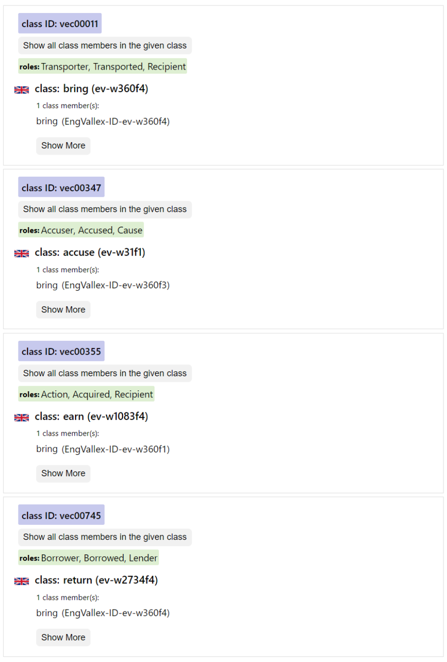


Class member (verb)	vallex ID (PDT or Eng)	Argument mapping			Links to external lexicons				Ex. WSJ sent.
		Complainer	Addressee	Complaint	Onto-notes	FrameNet	Propbank	Word-net	
complain	ev-w618f1	ACT	ADDR	PAT	1	Complaining	complain.01	#1	0020-s19
stěžovat si	v-w6521f2	ACT	ADDR	PAT|EFF					0118-s83
reptat	v-10975f2	ACT	LOC	PAT					2161-s22
grumble	ev-w1520f1	ACT	ADDR	PAT	1	Complaining	grumble.01	#1;#2	0934-s14
gripe	ev-w1508f1	ACT	ADDR(to)	PAT	--	Complaining	gripe.01	#1	0742-s53

Figure 1.1 The overall scheme of the SynSemClass lexicon and an example of a class ("complain-stežovat si") (Urešová et al., 2020).

In the rest of this section, we briefly summarize the initial annotation process of the ontology. We focus only on the steps that are relevant to us and omit the rest.

Urešová et al. (2017) built the first Czech and English version of the lexicon by semi-randomly choosing Czech verbs of various frequencies from the Prague Czech-English Dependency Treebank (PCEDT) (Hajič et al., 2012) and letting those be the names and seeds of new classes. They used automatic word alignment to get the English counterparts. Manual pruning and annotation followed. The PCEDT corpus contains an annotation layer relating to semantics and deep syntax, the so called tectogrammatical layer. Urešová et al. (2017) used this information to build the links to other resources as well as other linguistic annotation, which we will not delve into, as it does not affect our work.


class ID: vec00011

Show all class members in the given class

roles: Transporter, Transported, Recipient

class: bring (ev-w360f4)

1 class member(s):

bring (EngVallex-ID-ev-w360f4)

Show More

class ID: vec00347

Show all class members in the given class

roles: Accuser, Accused, Cause

class: accuse (ev-w31f1)

1 class member(s):

bring (EngVallex-ID-ev-w360f3)

Show More

class ID: vec00355

Show all class members in the given class

roles: Action, Acquired, Recipient

class: earn (ev-w1083f4)

1 class member(s):

bring (EngVallex-ID-ev-w360f1)

Show More

class ID: vec00745

Show all class members in the given class

roles: Borrower, Borrowed, Lender

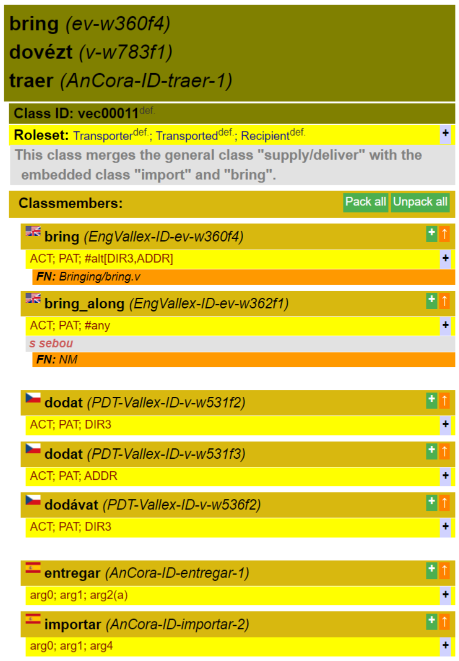
class: return (ev-w2734f4)

1 class member(s):

bring (EngVallex-ID-ev-w360f4)

Show More

(a) A lookup of the word ‘bring’ in the SynSemClass Search[1]tool shows that it belongs into multiple classes.


bring *(ev-w360f4)*
dovézt *(v-w783f1)*
traer *(AnCora-ID-traer-1)*

Class ID: vec00011^{def}

Roleset: Transporter^{def}; Transported^{def}; Recipient^{def}

This class merges the general class "supply/deliver" with the embedded class "import" and "bring".

Classmembers:

bring *(EngVallex-ID-ev-w360f4)*
ACT; PAT; #alt[DIR3,ADDR]
FN: *Bringing/bring.v*

bring_along *(EngVallex-ID-ev-w362f1)*
ACT; PAT; #any
s sebou
FN: *NM*

dodat *(PDT-Vallex-ID-v-w531f2)*
ACT; PAT; DIR3

dodat *(PDT-Vallex-ID-v-w531f3)*
ACT; PAT; ADDR

dodávat *(PDT-Vallex-ID-v-w536f2)*
ACT; PAT; DIR3

entregar *(AnCora-ID-entregar-1)*
arg0; arg1; arg2(a)

importar *(AnCora-ID-importar-2)*
arg0; arg1; arg4

(b) The class with ID vec00011 has several words associated with it as shown in the browse tool.[2]

Figure 1.2 Two different tools for the SynSemClass ontology. Note that both results are trimmed.

German was added by Urešová et al. (2022) by automatically word-aligning the English-German part of the ParaCrawl (Bañón et al., 2020) corpus. They decided on a set of English verbs, for which they found the most common German alignments. These were then manually filtered and annotated for correct classes. As they did not use a part-of-speech tagger, they could not distinguish between verb and noun forms of the verb (e.g., ‘to run’ vs. ‘a run’) and these cases had to be manually removed.

The Spanish part (Fernández-Alcaina et al., 2023) was built in a similar manner to German, using automatic word alignment on parallel data. Additionally, they used semantic information extracted from a Spanish lexical resource, AnCora (Taulé et al., 2008), to perform automatic filtering. Although AnCora contains only Spanish lemmas, it links to several English resources that are also linked in the SynSemClass ontology. If the automatic alignment created alignments for which semantic data was linkable to AnCora but the senses did not

[1]`https://lindat.mff.cuni.cz/services/SynSemClassSearch/`
[2]`https://lindat.cz/services/SynSemClass50/SynSemClass50.html`

match, the pairing was discarded. Unlike the German team, they had data tagged for part of speech.

There is a key difference between the initial phases of the SynSemClass annotation and the challenge of a new language presented here. In the initial phases of the SynSemClass annotation, more extensively annotated resources were available, including (automatically) word-aligned parallel corpora and manually annotated verb senses with respective valency dictionaries. In our case, the starting point is simply a sentence-aligned parallel corpus between a newly added language and a source language already annotated in the corpus, without any additional semantic information or a valency lexicon in the target language. Our language of choice is Korean. We describe Korean as well as our motivation behind choosing it in the next section.

We note that this work was done on SynSemClass 4.0[3]. The current version of SynSemClass is 5.0[4].

1.2 Korean language

We chose Korean as the language on which we test our methods for adding a completely new language into the ontology, as well as the overall design of the ontology to accommodate a non-European language. The author has a lower-intermediate knowledge of Korean, however, we would like to acknowledge that the work has not been checked by anyone either fluent in the language or with a sufficient linguistic background in it. We are confident that any errors made are of low significance and not damaging to the work as a whole. What follows is a brief introduction to the Korean language and its grammar.

Korean is an East Asian language spoken by more than 75 million people, most of whom live on the Korean peninsula. While being similar to Japanese in many aspects, the languages are generally regarded as not related and Korean is attributed its own language family, the Koreanic language family, whose only other member is the Jeju language, spoken by about 5,000 speakers on the island Jeju south of the Korean peninsula. Efforts trying to link Korean and Japanese are complicated by previous use of writing systems based on the Chinese writing system, which can obscure historic pronunciation. In 1443, Korean king Sejong the Great invented and later published a new, more phonemic, script. However, due to resistance from scholars and the upper classes, the script became widely adopted only in the second half of the 20th century.

Nowadays, the Korean script is called Hangul ⟨ 한글 ⟩ (with a different name used in North Korea). Depending on the way one counts, it uses 24-51 letters

[3]`https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-4746`
[4]`https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-5230`

(complex letters are considered to be made of basic letters, however they may have vastly different pronunciation, e.g., ㅏ/a/ + ㅣ/i/ = ㅐ/e/). The letters have mnemonic shapes, see figure 1.3 for an example.

Similar letters have similar shapes unlike most of the Latin alphabet (e.g., compare the Latin pair ⟨g⟩ ⟨k⟩ and the very roughly equivalent pair in Hangul ⟨ㄱ⟩ ⟨ㅋ⟩). Hangul letters are arranged into blocks, which consist of three positions for letters — initial, medial and an optional final position. For example, the letters ㅎ /h/ + ㅏ/a/ + ㄴ /n/ can form the block 한 /han/. For the rest of the work, we will use the Revised romanization system as reference, while also providing the Hangul counterpart.

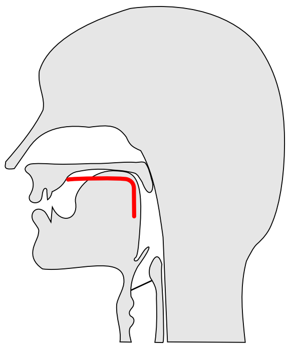


Figure 1.3 The Korean letter ⟨ㄱ⟩ /k/ is based on the placement of the tongue on the velum. Image adapted from Tavin and Nardog (2019).

Korean is an agglutinative language. Agglutination is the process of adding (often multiple) morphemes to a stem of a word, with the stem mostly remaining unchanged. To demonstrate the essence of this process and its difference from inflection, we will try to give an extreme example. It is not typical for a word to have this many morphemes, but it is not rare.

성공	seong-gong	success
적	jeog	"ful", affix meaning of having such character
이	i	be
었	eoss	past tense
다고	dago	quotation
말했다	malhaessda	said

성공적이었다고 말했다.
seong-gongjeogieossdago malhaessda.
'(He) said (he) was successful.'

In the above example, both subjects are omitted. Without context, it is unclear who said it and who was successful. This phenomenon is called pro drop (pronoun dropping). On top of dropping the subjects as is done in Slavic languages, Korean also allows for dropping of objects thanks to a phenomenon known as topic drop. The common phrase for 'I love you.' *saranghae* ⟨사랑해⟩ does not say who loves whom. An example conversation would be:

이	i	this
케이크	ke-ikeu	cake
좋아	joh-a	good
누가	nuga	who
만들었어	mandeur-eoss-eo	made

이 케이크 좋아. 누가 만들었어?
i keikeu joh-a. nuga mandeur-eoss-eo?
'This cake is good. Who made (it)?'

In Korean, the Subject-Object-Verb (SOV) word order is used, as opposed to English, Czech, German and most European languages, which usually use the Subject-Verb-Object (SVO) word order. This means that typically in a sentence, the subject comes first, followed by the object and the verb being the very last in the sentence.

나는	naneun	I
사람	saram	person
만났다	mannassda	met

나는 사람 만났다.
naneun saram mannassda.
'I met a person.'

The change in word order is also present elsewhere, most notably in relative clauses, which come before the word they are modifying.

바나나	banana	banana
준	jun	gave
그	geu	that
사람	saram	person
만났다	mannassda	met

바나나 준 그 사람 만났다.
banana jun geu saram mannassda.
'(I) met that person who gave (me) a banana.'

We hope this section was sufficient in giving a very basic overview of the Korean language and its distinction from European languages. We will provide more examples of Korean grammar as they become relevant for the following sections.

1.3 Related work

The task of predicting a SynSemClass ontology class might seem to be the same as word sense disambiguation, but we note that the two tasks are different. Navigli (2009) defines word sense disambiguation as selecting a single or more senses from a set of possible senses *for the given word*, that is, each word has a different set of possible senses. With the SynSemClass ontology, each verb can be assigned to any of the circa 1000 SynSemClass classes. We cannot limit the scope of the senses simply based on the lemma. That is for several reasons.

Most importantly, the SynSemClass project allows for idiomatic meaning. For example, the expression 'to lay the blame on someone' would be annotated the same as 'to blame' / 'to accuse' (Urešová et al., 2022, section 4.3). Even if this were not the case, we assume the ontology is still under construction, so the full scope of a word's meaning is not known, either because the given language is only partially annotated, or because we are in the beginning of the process of adding the language to the ontology.

In the case of designing a completely new hierarchy of classes or (word) senses on a given set of words from scratch, unsupervised clusterization techniques have also been suggested, e.g., Brown clustering (Brown et al., 1992). However, our situation differs from such a setting. The ontology is already partially constructed, containing about 1000 classes, and the classes are partially populated with Czech, English, German and Spanish words that express the concepts of those classes. Our challenge is to incorporate words that possibly express the same concept(s) from an incoming new language into one or more of the existing classes or decide that no such class is relevant for the new word.

2. Methodology

Our goal is to help the annotators of the SynSemClass ontology in adding a new language. We merely assume a parallel corpus, as opposed to the lexical resources and semantically annotated corpora available for the previously incorporated languages. To this end, we rely on a machine-learning classification model by Straková et al. (2023). For a given sentence, this model is able to predict the SynSemClass class for a marked verb. In this context, we evaluate two methods, both using this model, for adding a new language to the ontology. We describe these two methods later in this chapter. Using this model, we produce a preannotated language file. This file contains the verbs and their example usages collected from the corpus, grouped into SynSemClass classes. The task of the human annotators is then to manually accept or decline the automatically generated suggestions.

In this chapter, we describe the Korean data we use for our experiments and the methods we use to extract the annotation suggestions out of them. The details of our Python implementation are described in appendix B and the user documentation is in appendix A.[1]

2.1 Source data — parallel corpora

After careful consideration we chose to work with data[2] from Park et al. (2016), in particular their news corpus. The data consists of about 97k sentences. It was crawled from Yahoo! Korea and Joins.com[3] during 2010-2011. The authors provide no further information on the corpus, so the rest is just our observation. The data seem to be pieces of news articles from about 2000 to 2011. There are about 8 sentences per article and the sentences are ordered sequentially, that is in the order as they appeared in the text. Although no explicit article delimiters are included, the articles seem to be ordered chronologically, from the oldest to the newest. We believe the Korean text is original and the English translation is based on it. Sometimes, when English is translated into Korean, the Korean text can suffer from unusual syntactic structures and unnatural word choices. This phenomenon is known as translationese. Fortunately, in this corpus, both the Korean and English texts are of high quality. The sentence alignment is likely

[1]Source code: `https://github.com/petrkasp/synsemclass-pipeline`

[2]`https://github.com/jungyeul/korean-parallel-corpora/tree/master/korean-english-news-v1`

[3]The authors list the website as Joins CNN. According to Wikipedia, Joins.com is a (now defunct) website for the newspaper JoongAng Ilbo. We found no information about a link with CNN.

done automatically and there are many errors, both complete misalignments and ends or beginnings being cut off. Despite these shortcomings, we judged the corpus to be of the best quality among the parallel corpora available at the time.

We also considered several corpora[4] from OPUS (Tiedemann, 2009). Although the corpora are much bigger in size, they lack in quality. If we had needed a bigger corpus, our top choice would have been CCMatrix, both in terms of quality and its quantity of 19.4M sentences. The most common problem we see with the OPUS corpora are untranslated sentences aligned with each other. For example, this happens when a quote is left untranslated in the Korean text. Most of the texts also suffer from unnatural translation, although it is possible that this would have not been a problem for us given our focus on verbs.

2.2 UDPipe 2 — finding verbs

While it is possible that the ontology will expand to different word classes, so far, only verbs are considered. For finding the verbs in the data, we use UDPipe 2 (Straka, 2018)[5]. UDPipe 2 is a trainable natural language processing (NLP) pipeline capable of sentence segmentation, tokenization, part-of-speech tagging, lemmatization and dependency parsing. It is trained on the data from Universal Dependencies (UD; Nivre et al., 2016). Universal Dependencies is a project that collects treebanks, i.e., corpora annotated for syntactic structure, and provides a unified annotation style and harmonized tagsets for morphological analysis annotation and dependency parsing. It currently contains treebanks for 147 languages. UDPipe 2 provides models for 71 of those languages. The rest of the languages only have small treebanks that are insufficient for training a model.

UDPipe 2 takes in raw text and produces the requested annotations in the CoNNL-U format, which is the format used by Universal Dependencies. We use UDPipe 2 through the LINDAT UDPipe REST Service[6]. For further processing, we consider all words that are tagged `VERB`, with the exception of words that contain the 'to be' *i* ⟨이⟩ morpheme or end with a common adjective suffix *seureobda* ⟨스럽다⟩. We believe that words with this suffix are sometimes incorrectly tagged as verbs due to errors in the UD Korean treebanks.

[4]`https://opus.nlpl.eu/results/en&ko/corpus-result-table`
[5]`https://github.com/ufal/udpipe/tree/udpipe-2`
[6]`https://lindat.mff.cuni.cz/services/udpipe/`

2.3 Getting Korean lemmas

We also use UDPipe 2 for finding the lemmas of the verbs. We use the lemmas to group the occurrences to better understand what the correct SynSemClass classes of the lemma are. Unluckily to us, the Korean UD treebanks contain morphological analysis in the lemma field. To get the lemma of a verb, we take the first one or two morphemes and attach the *da* ⟨다⟩ suffix, which is the suffix of Korean lemmas for verbs and adjectives (sometimes called the dictionary form). If the second morpheme is *ha* ⟨하⟩ 'to do', *doe* ⟨되⟩ 'to become' or *shiki* ⟨시키⟩ 'to order', we replace it with *ha* ⟨하⟩ 'to do' and attach the lemma suffix *da* ⟨다⟩. Otherwise, we attach the *da* ⟨다⟩ suffix directly to the first morpheme.

To explain, Korean verbs can be divided into two groups, "native" verbs and *hada* ⟨하다⟩ verbs. The same holds for adjectives as they behave very similarly in terms of syntax. The "native" verbs form a closed class, new members are rarely added, albeit the class is very large compared to typical closed classes like prepositions. The stem of a "native" verb is a single morpheme, so we can attach the lemma suffix *da* ⟨다⟩ directly. Some examples of "native" verbs are *gada* ⟨가다⟩ 'to go', *mandeulda* ⟨만들다⟩ 'to make' or *gidarida* ⟨기다리다⟩ 'to wait'.

The *hada* ⟨하다⟩ verbs are an open class. When a foreign word is borrowed or a new concept needs to be named, a new verb enters the language as a *hada* ⟨하다⟩ verb. The class is not limited to borrowings and also contains native words (we use the term "native" for the other class because of a lack of a better term). *hada* ⟨하다⟩ verbs are formed by a noun followed by the *hada* ⟨하다⟩ suffix (where *da* ⟨다⟩ is the lemma suffix). While *hada* ⟨하다⟩ is a word on its own with the meaning of 'to do', the meaning of the verbs it creates is hardly "to do something" in the English sense, e.g., *malhada* ⟨말하다⟩ 'to speak, lit. speech+do', *haengboghada* ⟨행복하다⟩ 'to be happy, lit. happiness+do' (adjective).

The noun part and the *hada* ⟨하다⟩ part can often be separated, as seen on the example below.

준비+를	junbi+reul	preparation+accusative
안	an	not
했어	haess-eo	didn't do (past tense of *hada* ⟨하다⟩)

준비를 안 했어.
junbireul an haesseo.
(I) did not prepare.

However, not all words are separable, e.g., *haengboghada* ⟨행복하다⟩.

The *hada* ⟨하다⟩ part can be replaced with other suffixes. *doeda* ⟨되다⟩ 'to become' changes the word from active to passive voice and *shikida* ⟨시키다⟩

‘to order’ makes the meaning of ‘make someone do something’. We normalize these to ‘*hada* 〈하다〉’ to get more precise results. However, some nuance, and perhaps in some cases a wholly different meaning, can be lost.

We note that our lemmatization is not perfect and there are most likely exceptions that break it. Throughout our experiments, we tried to account for everything that came up, but we likely did not cover everything possible.

2.4 SimAlign — making word alignments

For annotation projection, word alignments are required. We construct these with SimAlign[7] (Jalili Sabet et al., 2020). SimAlign is a word alignment algorithm based on word embeddings (vector representations of words). It does not require any parallel data for training, only a model that generates the embeddings. This is done with a multilingual large language model (LLM). Embeddings are generated for both sentences, and then words with the closest embeddings are matched. For the matching, the authors propose several algorithms. In their experiments, their novel algorithm IterMax came out on top, so we use this algorithm for the matching. For the language model, we chose XLM-RoBERTa base (Conneau et al., 2020).

At the time of development, we believed SimAlign to be the state of the art, but improvements since have been made in the field (Lai et al., 2022; Dou and Neubig, 2021). In section 3.1.3, we conduct an experiment where we annotate the word alignment manually and discuss how a potential improvement in word alignment might influence the results.

2.5 Predicting SynSemClass classes

2.5.1 SynSemClass classification model

The SynSemClass classification model[8] (Straková et al., 2023) is a classifier that predicts the SynSemClass class for a marked verb in the input sentence. The verb has to be marked with a caret (^) to determine what verb should be predicted if multiple verbs are in the sentence. (For example, the word ‘marked’ in the following sentence has a mark, “we ^ marked the verb in this sentence.”) While Straková et al. (2023) only describe working with the Czech part of the ontology, they kindly provided us with a model that was fine-tuned on English, Czech and German data. The extra data for fine-tuning improve performance, reaching

[7]`https://github.com/cisnlp/simalign`
[8]`https://github.com/strakova/synsemclass_ml`

88.83% accuracy on the Czech test set, compared to 79.17% accuracy for the model fine-tuned solely on Czech. The combined accuracy on all test data is 86.02%, with 82.25% on English and 90.91% on German.

The foundation for the classification model is the pretrained large language model RemBERT (Chung et al., 2021). RemBERT is a multilingual masked language model trained on text from Wikipedia and Common Crawl. The RemBERT training data spans 110 languages.[9] Masked language modeling is an idea first introduced by Devlin et al. (2019). A masked language model (MLM) is trained by masking several random tokens within the text and having the model predict the masked tokens. Unlike classical n-gram language models and generative LLMs, e.g., GPT-style models, the training procedure provides MLMs with the ability to make predictions based on right context. This makes it a good candidate for fine-tuning as classifiers.

When modeling a problem with deep learning, many aspects of the problem need to be considered. One of them are the properties of the predicted data. Usually for classification, the assumption is that there is exactly one correct prediction (e.g., for digit classification, the input image contains exactly one digit 0–9). However, as the SynSemClass ontology is still in development, not all of the classes have been assigned yet. This means that some words do not belong to any class yet. One occurrence of a verb can also belong to multiple classes, if the meaning is not clear. Moreover, we want the model to help us identify suboptimal class definitions. Assume the model is often unsure between two specific classes. This might mean that the two classes are too similar and need merging. For these reasons, Straková et al. (2023) modeled the problem as circa 1000 separate binary classifications — one for each class. This leads to an overwhelming number of negative examples for each classifier (most classes have only a few examples, and even the numerous classes make up only a small portion of the data). Thus during training, instead of using the standard cross entropy loss (an error function used during training) a modification called focal loss (Lin et al., 2020) was used. According to the authors of focal loss, "focal loss focuses training on a sparse set of hard examples and prevents the vast number of easy negatives from overwhelming the detector during training."

2.5.2 Annotation projection

Cross-lingual annotation projection is a method that allows us to use a monolingual model to make automatic annotations for different languages other than the natively supported by the model. It relies on having parallel data with word

[9]The full list is available here
`https://github.com/google-research/google-research/tree/master/rembert`

	All data	Zero-shot
English	82.25%	63.59% (−18.66%)
Czech	88.83%	73.58% (−15.25%)
German	90.91%	44.44% (−46.47%)

Table 2.1 Accuracy of the SynSemClass classification models on the three languages available in the ontology. The results in the All data column are based on a single model fine-tuned on all three languages. The results in the Zero-shot column are each for a different model. These models saw no examples of the test language during fine-tuning and were fine-tuned only on the other two languages.

alignments. The word alignments can be manual or they can be automatically generated. We described how we make automatic word alignments in the previous section 2.4.

When working with annotation projection, we talk about two languages. The language supported by the classification model is called the *source* language. We want to create annotations for another language that we call the *target* language. Our source language is English and the target language is Korean. The monolingual model creates annotations in the parallel corpus on the source language text. Then the aligned words in the target language text are assumed to have the same annotation. This process can be quite noisy, due to both noise in the alignment and the classification model.

2.5.3 Zero-shot cross-lingual transfer

In our work, we assume no annotated data for our target language of Korean. However, as the classification model is based on a multilingual large language model, we can try to run it directly on the target language texts. This problem setup is called zero-shot cross-lingual transfer. In this setup, the model is faced with a task that it has not seen during training, but is expected to give reasonable predictions by generalizing the information obtained during pretraining and fine-tuning. See table 2.1 for the zero-shot accuracy of the models provided to us by Straková et al. (2023).

2.6 Evaluation data

To evaluate the two approaches described in the previous section, we need gold data. In order to be able to do this evaluation on Korean, we built a small corpus manually annotated for SynSemClass classes. Unlike the gold data for Czech, English and German extracted from the existing lexicon, we annotate every single verb in the sentences. This complete annotation alleviates some problems with the aggregation experiments we describe in section 3.2.

We start with the news corpus described in section 2.1 and use both annotation projection and zero-shot cross-lingual transfer to give us suggestions for the classes. We then go and remove misaligned sentences (15 in total), add verbs missed by the system, fix incorrect word alignment and select correct SynSemClass classes for each of the verbs, all manually by hand. In order to give the best chances to annotation projection, we also denote how accurate our manual alignment is. In many cases, no good alignment is possible. This happens because the translation is not literal. For example, in one of the sentences, the phrase *shilbaga ikkeuneun dang-ui bulg-eun gisbal* ⟨실바가 이끄는 당의 붉은 깃발⟩ 'the red flag of the party led by Silva' is translated as 'his [Silva's] party's red flag'.

In this manner, we manually annotate 131 verbs in 39 sentences. Selecting the correct SynSemClass class out of the total of 884 semantic classes was a very difficult and time consuming task, especially since the SynSemClass search tool was not available at the time.

The format of the pregenerated suggestions and finished annotated parallel corpus is described in appendix C.

3. Experiments & results

In this chapter we describe the experiments we conducted with the toolchain composed of the tools introduced in the previous chapter. We also present the results of the experiments and undertake an analysis of the source of the errors made by the toolchain.

In our main experiment, we determine whether annotation projection or zero-shot cross-lingual transfer performs better on our task of creating suggestions for a semantic ontology. We also give statistical significance for this finding. After concluding that zero-shot cross-lingual transfer performs better, we explore the shortcomings of annotation projection. We find that a fundamental limitation lies in the translation itself and that it often has a meaning too different to be useful for determining the semantic class on the other language side.

In our second experiment, we determine the best parameters for our toolchain with the zero-shot cross-lingual transfer approach. Specifically, we explore what the best way to aggregate the predictions is.

3.1 Is annotation projection or zero-shot transfer better?

We explore and compare the performance of annotation projection and zero-shot cross-lingual transfer. We conduct this experiment on the individual predictions, not on the aggregates. This allows us to explore the reasons behind the errors and better understand them.

3.1.1 Metrics

Our focus is on two metrics — recall and precision. The two metrics are widely used across computer science, and even in other fields (perhaps under different names). As there is some nuance in their usage, we dive into their definitions a little deeper.

Recall is fundamentally defined as the ratio of correct predictions and the maximum number of correct predictions achievable. Precision is the ratio of correct predictions and the total number of predictions. In classification (into two classes, one of them being the positive class), recall and precision are defined as follows:

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

In information retrieval, recall is the ratio of the retrieved documents that are relevant and the total number of relevant documents, while precision is defined as the ratio of the retrieved documents that are relevant and the total number of retrieved documents.

Our system has characteristics of both. While we indeed classify words into semantic classes, we need to find them in the text first. Even after we find the verb, annotation projection can in some sense reject to classify the verb if there is nothing aligned with it. Thus for a verb to be a correct prediction, three things need to happen:

1. UDPipe 2 has to correctly classify it as a verb.
2. SimAlign needs to align it.
3. The SynSemClass classification model needs to classify it into the correct class.

Step 2 is not needed with zero-shot cross-lingual transfer and we later show that this is a substantial advantage.

Thus for our purposes, we define recall as the ratio of correct predictions as described above and the total number of gold verbs. Precision is the ratio of correct predictions and the number of verbs found by UDPipe 2, with annotation projection, they also need to be aligned.

3.1.2 Zero-shot cross-lingual transfer wins as a system

We now measure the performance of the two methods when composed into the whole system. In section 3.1.4, we measure the performance of each individual component.

Our system consists of taking the raw sentences in the corpus (i.e., without the verb marks) and having UDPipe find the verbs in the Korean text. With zero-shot cross-lingual transfer, we predict the SynSemClass classes directly on the discovered Korean verbs. With annotation projection, we first use SimAlign to find the counterparts on the English side and make the predictions on the aligned English words. Here, many adjustments that could improve the precision of the

	Recall	Precision
Annotation projection	0.37	0.41
Zero-shot transfer	**0.54**	**0.54**
p-value	0.00040	0.00110

Table 3.1 Performance of the two methods used in the overall system with automatic verb identification and word alignment. The p-value indicates the probability of zero-shot transfer outperforming annotation projection by chance.

alignments are possible, e.g., filtering the alignments for part-of-speech, finding the verbs on the English side and doing the alignments vice-verse, or somehow crosschecking these two approaches. To account for these possible changes, in section 3.1.3, we conduct an oracle experiment where we use manual gold word alignments.

The results are presented in table 3.1. On our data, zero-shot transfer significantly outperforms annotation projection in both metrics. As the amount of data we have is rather small, we also perform a statistical test. We chose to do a permutation test on sentences with Monte Carlo sampling with 10k samples. The results are in table 3.1. They indicate that the findings are statistically significant.

3.1.3 Non-verbatim translation is a fundamental limitation

As we described in section 2.6, during the manual annotation of alignment in the Korean corpus, we encountered many examples of verbs that did not have a proper counterpart on the English side. Thus we divide the gold manual alignments into three groups. In the *none* group, no alignment is possible due to different phrasing and non-literal translation. The *dubious* alignment group are verbs that had a verb of a wholly different meaning or not a verb at all as their aligned counterpart. The rest of verbs fall into the *good* alignment group.

We perform experiments on the portions of the data that contain only the given quality alignment group or better. For both zero-shot cross-lingual transfer and annotation projection, we remove UDPipe 2 and instead use our annotations directly. For annotation projection, we conduct the experiments both with and without the automatic word aligner. Note that is experiment also assumes perfect sentence alignment throughout. The results of the experiment, as well as how much of the data is covered by each quality requirement, are presented in table 3.2. In this experiment, we simply use accuracy and omit recall or precision when they are equivalent to accuracy.

We see that the good alignments make up only 73% of the data. We consider this to be a significant limitation for the possible quality of annotation projection. The quality of the predictions increases as the quality of the alignment

	Alignment type		
	None	Dubious	Good
Coverage	100%	83%	73%
Zero-shot			
Accuracy	**66%**	**62%**	62%
Annotation projection			
manual alignments			
Accuracy	51%	61%	**66%**
automatic alignments			
Accuracy	46%	52%	57%
Precision	55%	58%	61%
Word alignment			
Accuracy	60%	62%	66%
Precision	62%	69%	71%
Rejection rate	16%	10%	6%

Table 3.2 The quality of predictions and automatic word alignments under each type of alignment. We split the best possible word alignment into three categories introduced in section 2.6. The coverage row describes how much of the data fits into this category or any better. We perform the experiments on this portion of the data. Where either precision or recall is not specified, its meaning is the same as accuracy. The word alignment section describes only the performance of the word aligner, not any predictions. Rejection rate is the ratio of words that were not aligned with anything.

increases. The predictions on our manual alignment outperform the predictions on the automatically aligned words and the performance increases in accordance with our alignment groups. This is in accord with previous literature. Behzad et al. (2023) recently reported on how manual correction of word alignment quality can improve performance on the downstream task.

The experiment also reveals that the accuracy of the model in the zero-shot setting on Korean is 66%. Interestingly, it is highest when all data are included and falls when we omit the sentences that had no alignment to English. We hypothesize that common fixed expressions are mostly translated non-verbatim, but we did not investigate this idea further. Annotation projection also achieves 66% in its ideal setting with ideal word alignments. However, if automatic alignment is used, it again trails behind.

3.1.4 Error contribution of each component

In table 3.3, we summarize the performance of each component. When we reported the results of the whole system in table 3.1, the results reflected that the system can reject some input verb early on. This verb does not proceed to the next components, e.g., UDPipe 2 fails to find a specific verb. To truly see the effect of each individual component, we give them the gold results of the previous step as input. For this reason, the product of the performances of each individual component does not equal the performance of the whole system. Every component is penalized for failing on the same hard example, while the system could reject it early on.

UDPipe 2 — finding verbs	
Recall	75%
Precision	76%
Extra verbs per sentence	0.85
Translation	
Verbs alignable	73%
SimAlign — Word alignment	
Accuracy	66%
Precision	71%
SynSemClass classification model	
Korean: Accuracy	66%
English: Accuracy	66%

Table 3.3 Performance of each component in separation. We also include the translation as a "component", as we believe it to be a integral part of annotation projection and a likely point of failure. The reported values are independent of each other — they are measured in ideal conditions for that component (gold outputs of previous steps). Thus, if multiplied, they do not equal the results in table 3.1. Here, each component is penalized for failing on the same hard examples the previous components failed on.

As we have seen, the accuracy of annotation projection is hindered by non-verbatim translation. More loose styles of translation are very common for high quality translations such as in our corpus. For this reason, we also think of translation as an inherent component of the annotation projection. Another crucial step we do not take into account is sentence alignment. We only work with examples that are properly aligned on the sentence level. During the annotation of our evaluation data, we discarded all sentences with improper sentence alignment.[1] Of the 54 sentences considered, we kept 72% of them, however, some

[1]We kept a few pairs where the source English sentence had extra words compared to the

were discarded due to other reasons than bad alignment, e.g., a lack of any verbs.

The results show that, on our data, the classification model performs roughly the same on English and Korean. This might be due to the Korean text being original and the English text being the translation, thus the English translation can be possibly slightly unnatural or perhaps just different than natural English text. We cannot rule out that the results would be different for translation in the opposite direction. However, in our setting, the translation and automatic alignment (both on words and likely on sentences) introduce extra error.

3.2 Best aggregation parameters

We established that zero-shot cross-lingual transfer performs better than annotation projection. Now our goal is to produce the annotation suggestions for the SynSemClass ontology using the preferred method. We need to make the suggestions as aggregations of the the individual predictions, both for practical and technical reasons. The suggestions need to be of high quality and the optimum trade-off between the number of displayed and still relevant suggestions must be found in order to save the costly time of annotators. Here we try to determine what are the most efficient ways to aggregate the predictions for specific precision/recall requirements by studying the precision-recall curves.

An annotation suggestion is a class-lemma pair and each suggestion appears at most once. This is given by the language-specific SynSemClass file format in which the suggestions are to be stored. The SynSemClass classification model gives us probabilities[2] how likely each example is to be of the given sense and we need to aggregate them into these suggestions. As we want to evaluate the best aggregation strategy as well as an optimal threshold on the final aggregation score, we need gold data of accepted and rejected suggestions. For this purpose we leverage our existing evaluation data. Alongside our manually annotated Korean–English parallel corpus, we also have the examples extracted from the existing ontology.

Both of these datasets are gold annotated sentences, not samples of rejected and accepted annotation suggestions, but under the right assumptions, we can use them to decide whether an automatic suggestion should be rejected or accepted. We generate the suggestions using the text from the corpus and evaluate them using the following rules:

1. If the suggestion contains a lemma that is not annotated in our gold data, the suggestion is ignored, i.e., considered neither correct nor wrong.

Korean target sentence, as this did not prevent word alignment.

[2]The term probability might not be completely accurate here, as the values do not sum to 1 due to the way the problem is modeled as explained in section 2.5.1.

2. The suggestion is accepted if there exists an example in the corpus that supports that suggestion. This means there must be a sentence in the corpus that contains the same lemma with the class the suggestion is proposing.
3. Otherwise, we reject the suggestion.

We need to apply step 1 because the examples from the ontology contain unannotated verbs. The sentences showcase a single class on a single verb, but a lot of the sentences contain other verbs that are not annotated. Unfortunately, this approach introduces several problems:

a) If the system overgenerates, e.g., incorrectly labels a word as a verb, it is ignored. This can slightly inflate the measured performance.

b) A single lemma can have different meanings. Suppose that a lemma is present in the corpus with two meanings, but only one of them is annotated. If the system finds the unannotated verb and makes a suggestion based on it, it is incorrectly punished. This can lower the measured performance.

c) A slightly more nuanced version of problem b) — even if all verbs in the corpus were annotated, we still do not cover every correct suggestion. The system might mislabel a verb with a class that is not correct in the given context, but might be correct in another context. As we also take into account other predictions than the top one, this might introduce some false errors.

Fortunately, problems a) and b) do not apply to our Korean–English corpus, as we have manually labeled every verb. On the other hand, the corpus is very small, so the effect of aggregation is barely observable.

We conduct the experiments and take the described problems into account when interpreting the results. On the Korean dataset, we do not apply the filtering rule (rule 1) as it is not needed as explained above. In all experiments, we generate precision-recall curves and compare the results. For easier comparison, we set the ranges of all figures to a fixed value, so some of the figures have a lot of empty padding. All experiments in this section use zero-shot cross-lingual transfer. To better understand the results, we also provide the the dataset sizes in table 3.4.

3.2.1 Best aggregation function

We consider four aggregation functions: average, maximum, summation and a probabilistically motivated function we shortly explain. As their input, the functions take all predictions on a specific lemma and a specific SynSemClass class.

Language	Sentences
Partially annotated	
English	7896
Czech	8942
German	985
Fully annotated	
Korean	39

Table 3.4 Dataset sizes in sentences.

On each verb example, we consider all 884 predicted probabilities for each class, not just the top one. With the averaging function, we also conduct further experiments how limiting the tail predictions affects the results. For the maximum aggregation function, this limiting would have little effect due to it deciding only on the largest value for the given lemma-class pair.

The probabilistic function assumes that the suggestion is correct, if at least one prediction is correct, and that the predictions are independent. The function is then defined as:

$$prob(X) = 1 - \prod_i 1 - prob(X_i)$$

Figure 3.1 shows the results. The German and Korean datasets do not show much difference between the functions. We believe this is due to their small sizes. Due to this, the effects of aggregation are hardly seen. On English and Czech, we can observe that for high precision (high thresholds, small recall), average is dominating. However, for high recall (small threshold, small precision), maximum wins. We explain this intuitively — for a high recall, every high probability prediction is worth considering, even though there are a lot of small value predictions for this class-lemma pair.

3.2.2 Preaggregation filtering

In the previous experiment, we considered all 884 values, one for each class, predicted by the classifier on each mention of a verb. This includes a lot of very small values around and below 1% that influence the results. We test whether it is effective to filter those small values before aggregation. There two ways to do this — setting a threshold for probability, e.g., ignoring everything below 5%, or only considering the top K predictions for each lemma. The results for the thresholding approaches are displayed in figure 3.2 and the results for top-K are in figure 3.3.

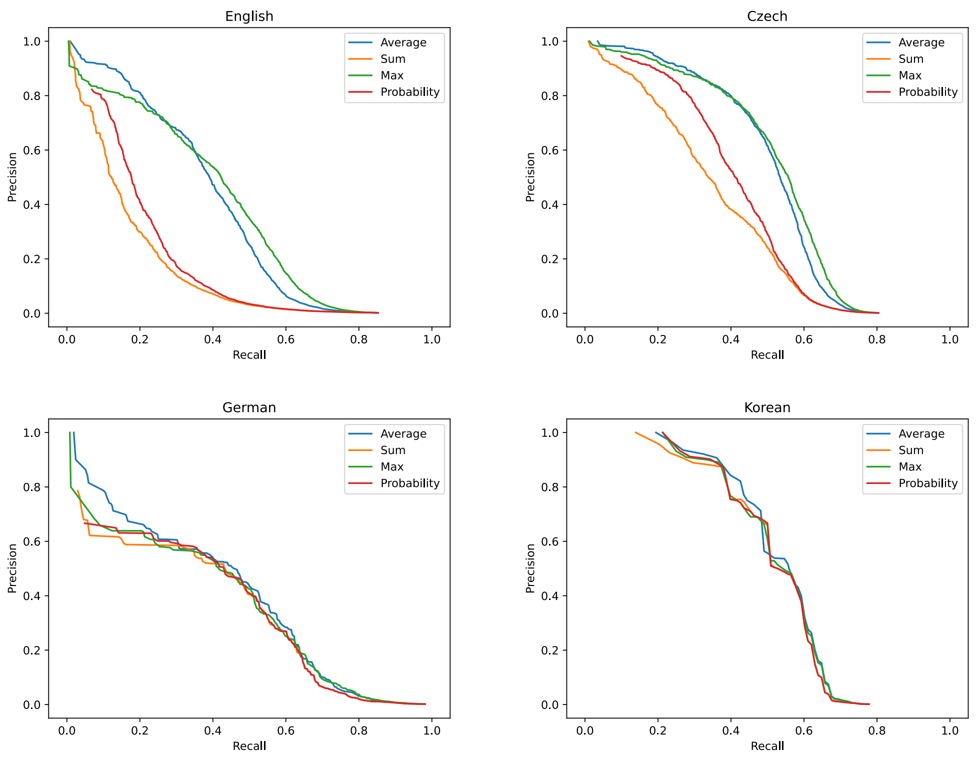


Figure 3.1 Precision-recall curves for various aggregation functions.

We see trends similar to the results for aggregation functions. The curves for German and Korean, as well as the high precision parts of the curves for English and Czech behave the same. The curves suggest that it is better not to limit the input, i.e., no threshold and all predictions, not just top K predictions. We see this as intuitive, if we take more input, we get more precise results and suggestions that are not certain are pushed down. For the high recall parts of the curve, a good threshold seems to be around 3% and the best K, for top K predictions, seems to be 3. At the same time, the maximum aggregation function performs very similarly to these two settings.

3.2.3 Possible limitations

We now consider the effects of the problems described earlier and think of how they affect the results. In a large corpus, we can encounter polysemantic verbs that often appear with two or more senses. It is possible that one of the senses is in our gold data while the others are not, even though they are all correct. As we aggregate a large number of predictions, the scores of these polysemantic verbs

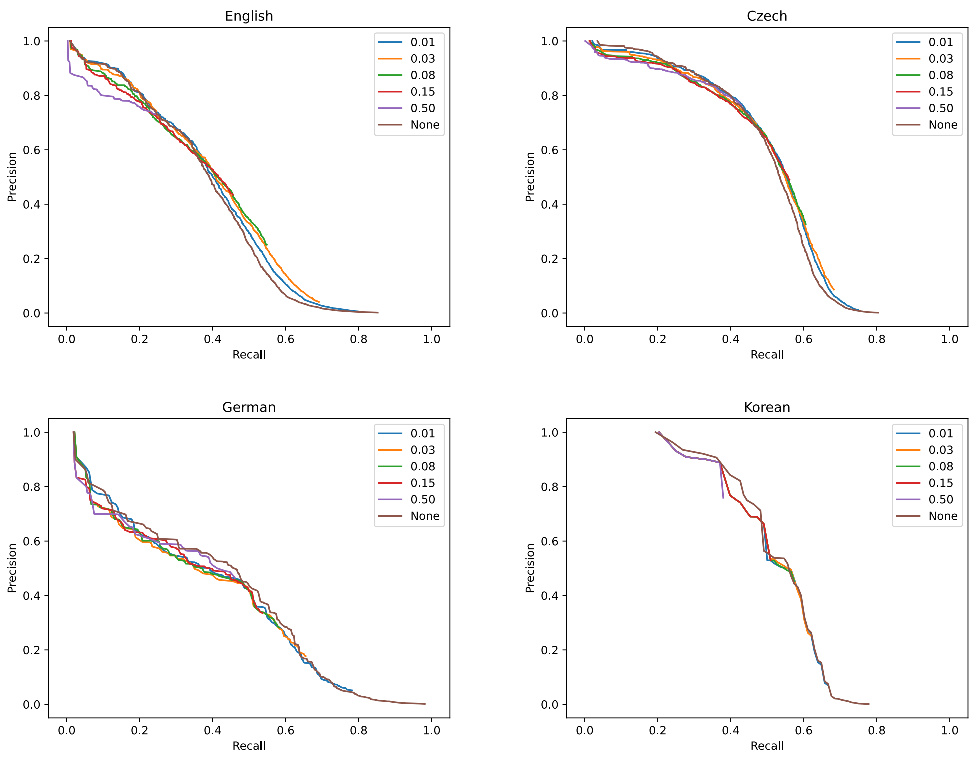


Figure 3.2 Precision-recall curves for filtering predictions with a probability threshold. Some curves suddenly end as they do not have a high recall portion. This is because the setting inherently only shows high precision results and completely filters out the unlikely suggestions.

are lowered. This can give higher measured precision to the final suggestions, as the other senses of the polysemantic verbs are falsely labeled as errors, but with more predictions, their scores are lowered. This is in agreement with our observation that the aggregation function of average with no filtering performs the best for high precision of the suggestions. However, we see that average with no filtering performs well on Korean, too, and we think that Korean does not suffer from the described problems. The difference between average with no filtering and the other approaches is not as prominent there, but it is still clearly dominating.

As for using maximum or filtering for higher recall, we would expect the opposite with the previous line of reasoning. Moreover, the actual difference might be even larger than we measured and using maximum and filtering might be more beneficial than our results suggest. While we lay some groundwork for the optimal parameters, we believe that it would be best to do the measurements

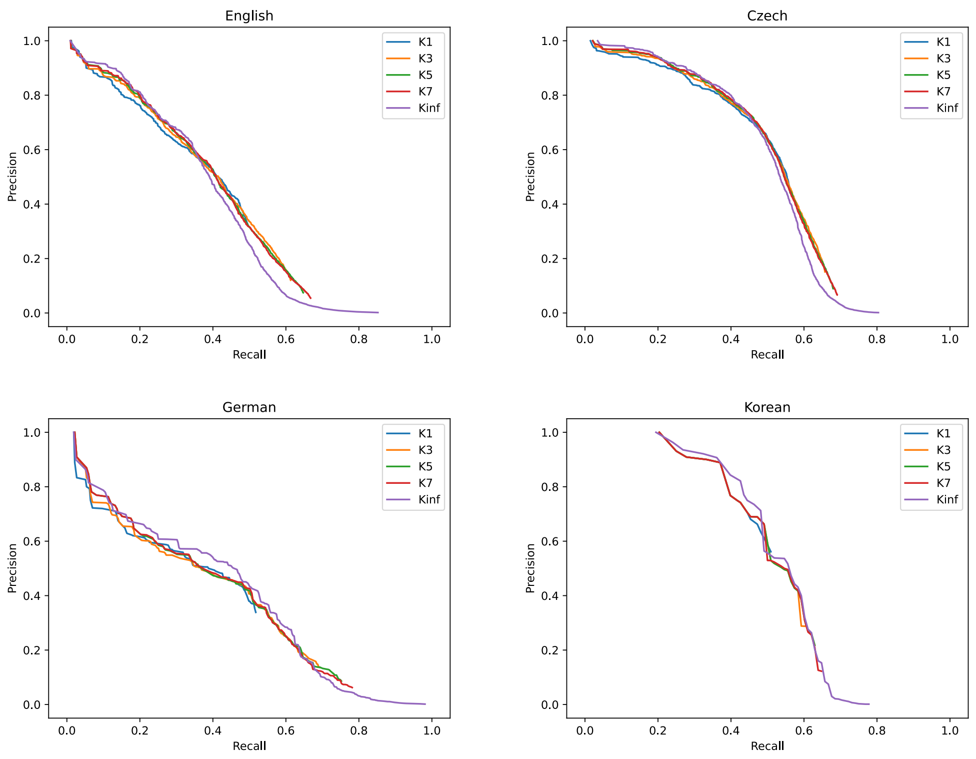


Figure 3.3 Precision-recall curves for filtering predictions by taking only the top K values into account for each lemma. Kinf means all values. Same as with figure 3.2, some curves end abruptly and do not have a high recall portion. This is because the setting inherently only shows high precision results and completely filters out the unlikely suggestions.

again when the annotation phase starts and more data is available.

Conclusions

We implemented and optimized a toolchain for creating annotation suggestions for the semantic ontology SynSemClass. Unlike the previous work on the ontology, we did not have access to any lexical or semantic resources. We evaluated two approaches — annotation projection that uses a parallel corpus to project predicted classes from a source language to the target language, and zero-shot cross-lingual transfer that relies on the ability of a machine learning model to generalize to a language it was not trained on. For the purpose of the experiments, we developed and manually annotated for semantic classes a small Korean–English corpus. We found that zero-shot cross-lingual transfer performs significantly better both in terms of recall and precision and verified this statistically.

In order to generate high-value annotation suggestions that are likely to be accepted by annotators, we compared various aggregation approaches using precision-recall curves. We found that taking the average of all predictions including all predictions for each example and not limited to the top one works well. In scenarios where high recall is desired, it seems beneficial to either use maximum instead of average, or take only the three best predictions, or only consider predictions with probability higher than 3%.

We also analyzed why zero-shot cross-lingual transfer performs better. We found that non-verbatim translation poses a fundamental problem as sometimes the text is rephrased in such a way that the verb disappears or changes meaning significantly. All of the additional steps introduce cascading errors while the raw predictions are of the same quality on either language, even if not specifically trained on it.

Bibliography

Bañón, Marta, Pinzhen Chen, Barry Haddow, Kenneth Heafield, Hieu Hoang, Miquel Esplà-Gomis, et al. (July 2020). “ParaCrawl: Web-Scale Acquisition of Parallel Corpora”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault. Online: Association for Computational Linguistics, pp. 4555–4567. DOI: `10.18653/v1/2020.acl-main.417`. URL: `https://aclanthology.org/2020.acl-main.417`.

Behzad, Shabnam, Seth Ebner, Marc Marone, Benjamin Van Durme, and Mahsa Yarmohammadi (July 2023). “The Effect of Alignment Correction on Cross-Lingual Annotation Projection”. In: *Proceedings of the 17th Linguistic Annotation Workshop (LAW-XVII)*. Ed. by Jakob Prange and Annemarie Friedrich. Toronto, Canada: Association for Computational Linguistics, pp. 244–251. DOI: `10.18653/v1/2023.law-1.24`. URL: `https://aclanthology.org/2023.law-1.24`.

Brown, Peter F., Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer (1992). “Class-Based *n*-gram Models of Natural Language”. In: *Computational Linguistics* 18.4, pp. 467–480. URL: `https://aclanthology.org/J92-4003`.

Chung, Hyung Won, Thibault Fevry, Henry Tsai, Melvin Johnson, and Sebastian Ruder (2021). “Rethinking Embedding Coupling in Pre-trained Language Models”. In: *International Conference on Learning Representations*. URL: `https://openreview.net/forum?id=xpFFI_NtgpW`.

Cinková, Silvie, Eva Fučíková, Šindlerová, and Jan Hajič (2021). *EngVallex - English Valency Lexicon 2.0*. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University. URL: `http://hdl.handle.net/11234/1-3526`.

Conneau, Alexis, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, et al. (July 2020). “Unsupervised Cross-lingual Representation Learning at Scale”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault. Online: Association for Computational Linguistics, pp. 8440–8451. DOI: `10.18653/v1/2020.acl-main.747`. URL: `https://aclanthology.org/2020.acl-main.747`.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (June 2019). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language*

Technologies, Volume 1 (Long and Short Papers). Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

Dou, Zi-Yi and Graham Neubig (Apr. 2021). "Word Alignment by Fine-tuning Embeddings on Parallel Corpora". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Ed. by Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty. Online: Association for Computational Linguistics, pp. 2112–2128. DOI: 10.18653/v1/2021.eacl-main.181. URL: https://aclanthology.org/2021.eacl-main.181.

Fernández-Alcaina, Cristina, Eva Fučíková, Jan Hajič, and Zdeňka Urešová (2023). "Spanish Synonyms as Part of a Multilingual Event-Type Ontology". In: *Journal of Linguistics/Jazykovedný časopis* 74.1, pp. 153–162. DOI: doi:10.2478/jazcas-2023-0033. URL: https://doi.org/10.2478/jazcas-2023-0033.

Hajič, Jan, Eva Hajičová, Jarmila Panevová, Petr Sgall, Silvie Cinková, Eva Fučíková, et al. (2012). *Prague Czech-English Dependency Treebank 2.0*. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University. URL: http://hdl.handle.net/11858/00-097C-0000-0015-8DAF-4.

Jalili Sabet, Masoud, Philipp Dufter, François Yvon, and Hinrich Schütze (Nov. 2020). "SimAlign: High Quality Word Alignments without Parallel Training Data using Static and Contextualized Embeddings". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*. Online: Association for Computational Linguistics, pp. 1627–1643. URL: https://www.aclweb.org/anthology/2020.findings-emnlp.147.

Kingsbury, Paul and Martha Palmer (May 2002). "From TreeBank to PropBank". In: *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*. Ed. by Manuel González Rodríguez and Carmen Paz Suarez Araujo. Las Palmas, Canary Islands - Spain: European Language Resources Association (ELRA). URL: http://www.lrec-conf.org/proceedings/lrec2002/pdf/283.pdf.

Lai, Siyu, Zhen Yang, Fandong Meng, Yufeng Chen, Jinan Xu, and Jie Zhou (Dec. 2022). "Cross-Align: Modeling Deep Cross-lingual Interactions for Word Alignment". In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, pp. 3715–3725. URL: https://aclanthology.org/2022.emnlp-main.244.

Lin, Tsung-Yi, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár (2020). "Focal Loss for Dense Object Detection". In: *IEEE Transactions on Pattern Anal-*

ysis and Machine Intelligence 42.2, pp. 318–327. DOI: 10.1109/TPAMI.2018.2858826.

Navigli, Roberto (2009). "Word sense disambiguation: A survey". In: *ACM Comput. Surv.* 41.2. ISSN: 0360-0300. DOI: 10.1145/1459352.1459355. URL: https://doi.org/10.1145/1459352.1459355.

Nivre, Joakim, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, et al. (May 2016). "Universal Dependencies v1: A Multilingual Treebank Collection". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. Ed. by Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, et al. Portorož, Slovenia: European Language Resources Association (ELRA), pp. 1659–1666. URL: https://aclanthology.org/L16-1262.

Park, Jungyeul, Jeen-Pyo Hong, and Jeong-Won Cha (Oct. 2016). "Korean Language Resources for Everyone". In: *Proceedings of the 30th Pacific Asia Conference on Language, Information and Computation: Oral Papers*. Seoul, South Korea, pp. 49–58. URL: https://aclanthology.org/Y16-2002.

Ruppenhofer, Josef, Michael Ellsworth, Myriam Schwarzer-Petruck, Christopher R Johnson, and Jan Scheffczyk (2016). *FrameNet II: Extended theory and practice*. Tech. rep. International Computer Science Institute.

Straka, Milan (Oct. 2018). "UDPipe 2.0 Prototype at CoNLL 2018 UD Shared Task". In: *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Brussels, Belgium: Association for Computational Linguistics, pp. 197–207. DOI: 10.18653/v1/K18-2020. URL: https://www.aclweb.org/anthology/K18-2020.

Straková, Jana, Eva Fučíková, Jan Hajič, and Zdeňka Urešová (July 2023). "Extending an Event-type Ontology: Adding Verbs and Classes Using Fine-tuned LLMs Suggestions". In: *Proceedings of the 17th Linguistic Annotation Workshop (LAW-XVII)*. Ed. by Jakob Prange and Annemarie Friedrich. Toronto, Canada: Association for Computational Linguistics, pp. 85–95. DOI: 10.18653/v1/2023.law-1.9. URL: https://aclanthology.org/2023.law-1.9.

Taulé, Mariona, M. Antònia Martí, and Marta Recasens (May 2008). "AnCora: Multilevel Annotated Corpora for Catalan and Spanish". In: *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*. Ed. by Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, et al. Marrakech, Morocco: European Language Resources Association (ELRA). URL: http://www.lrec-conf.org/proceedings/lrec2008/pdf/35_paper.pdf.

Tavin and Nardog (2019). *Voiceless velar plosive*. This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit https://creativecommons.org/licenses/by-sa/4.0/deed.en. Our derivative is also licensed as such. URL: https://commons.wikimedia.org/wiki/File:Voiceless_velar_plosive.svg.

Tiedemann, Jörg (Jan. 2009). "News from OPUS—A Collection of Multilingual Parallel Corpora with Tools and Interfaces". In: vol. 5, pp. 237–248. ISBN: 9789027248251. DOI: 10.1075/cilt.309.19tie.

Urešová, Zdeňka, Eva Fučíková, Eva Hajičová, and Jan Hajič (2017). "Syntactic-Semantic Classes of Context-Sensitive Synonyms Based on a Bilingual Corpus". In: *Proceedings of 8th Language and Technology Conference*. Poznań, Poland: Fundacja Uniwersytetu im. Adama Mickiewicza w Poznaniu, pp. 201–205. ISBN: 978-83-64864-94-0. URL: https://ufal.mff.cuni.cz/~hajic/2017/docs/LTC_17.pdf.

Urešová, Zdeňka, Eva Fučíková, Eva Hajičová, and Jan Hajič (May 2020). "SynSemClass Linked Lexicon: Mapping Synonymy between Languages". English. In: *Proceedings of the 2020 Globalex Workshop on Linked Lexicography*. Ed. by Ilan Kernerman, Simon Krek, John P. McCrae, Jorge Gracia, Sina Ahmadi, and Besim Kabashi. Marseille, France: European Language Resources Association, pp. 10–19. ISBN: 979-10-95546-46-7. URL: https://aclanthology.org/2020.globalex-1.2.

Urešová, Zdeňka, Eva Fučíková, and Jana Šindlerová (Apr. 2016). "CzEngVallex: a Bilingual Czech-English Valency Lexicon". In: *The Prague Bulletin of Mathematical Linguistics* 105, pp. 17–50. DOI: 10.1515/pralin-2016-0001.

Urešová, Zdeňka, Karolina Zaczynska, Peter Bourgonje, Eva Fučíková, Georg Rehm, and Jan Hajič (June 2022). "Making a Semantic Event-type Ontology Multilingual". In: *Proceedings of the Thirteenth Language Resources and Evaluation Conference*. Ed. by Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, et al. Marseille, France: European Language Resources Association, pp. 1332–1343. URL: https://aclanthology.org/2022.lrec-1.142.

A. Using the pipeline

The pipeline is run through the `main.py` script (located in the `scripts` folder). The pipeline uses corpora in one or two languages. The *target* language is the one that is yet to be added to the ontology and which we want to pre-annotate. The second language is the *source* language. It is used to provide possibly better suggestions for annotations of the target language. The source language is usually English, but not necessarily.

The `main.py` script accepts the following arguments:
(See `.vscode/launch.json`[1] for examples.)

- `task`: chooses the task, possible values are
  - `corpus`: Generates a corpus with suggested SynSemClass annotations.
  - `ssc`: Generates a SynSemClass (SSC) language-specific file with suggested annotations.
  - `evaluate`: Evaluates the quality of the pipeline suggestions using given gold data.
  - `pred-matrix`: Creates and saves a compact representation of all the predictions for use in experiments.

The following arguments are used by all tasks:

- `udpipe-service`: URL to the UDPipe service
  (default: `https://lindat.mff.cuni.cz/services/udpipe/api`)

- `udpipe-model-target` (required): UDPipe model for the target language, a two letter language code can be used to select a model automatically.

- `udpipe-model-source`: UDPipe model for the source language.

- `ssc-pred-model` (required): path to the model used for SynSemClass predictions

- `lemmatization`: path to the script to perform lemmatization. The script should contain a function called `lemmatize` that takes a word from the `ufal.udpipe` package and returns the lemma. If not provided, the lemma returned by UDPipe 2 is used.

[1]`https://github.com/petrkasp/synsemclass-pipeline/blob/main/.vscode/launch.json`

- `verb-filter`: path to the script to perform verb filtering. The script should contain a function called `verb_filter` that takes a word from the `ufal.udpipe` package and True iff the word should be considered for the pipeline. If not provided, all words marked as `VERB` by UDPipe 2 will be used.

An example lemmatization and verb filtering script can be found in `scripts/ko_filter_lemma.py`.[2]

The following arguments are used for the corpus and ssc tasks:

- `output`: path to where the output corpus/SynSemClass file will be produced.

Either of these should be provided:

- `target-corpus`: Plain text corpus of the target language. (Sentence per line)

- `source-corpus`: Corpus parallel to the target corpus in the source language.

Or:

- `tmx-corpus`: A parallel corpus in the tmx format.

The following arguments are used for the ssc task:

- `lang-name`: Three letter language code that is inserted into the SynSemClass file.

- `corpref`: A short name of the corpus to be inserted into the SynSemClass file.

- `member-threshold`: The threshold for members. Members with lower scores will not be put into the SynSemClass file.

- `aggregation`: The aggregation function to use.

- `predict-max-k`: Only top K predictions for each example will be taken into account.

- `example-threshold`: Only predictions with higher probability will be taken into account.

[2]`https://github.com/petrkasp/synsemclass-pipeline/blob/main/scripts/ko_filter_lemma.py`

- `example-count`: Number of examples to show for each member in the SynSemClass file.

The following argument is used for the evaluation task:

- `evaluation-corpus`: Golden data used as evaluation in the format as produced by the corpus task (see below for description).

B. Implementation

Our implementation[1] is written in Python and uses the methods and their respective libraries as described in chapter 2. For development, we used Visual Studio Code (VS Code). We developed the tool remotely on the Institute of Formal and Applied Linguistics' Artificial Intelligence Cluster.[2]

The dependencies are managed by the Python package-management system pip, except for the SynSemClass classification model, which is loaded as a subrepository.[3] The SynSemClass classification model also has its own dependencies managed by pip. Our code is located in the `scripts` folder. Here is an overview of the files:

- `main.py` — the entry point of the toolchain. The bulk of the logic is contained here.
- `ko_filter_lemma.py` — contains the definitions of how lemmatization and verb filtering should be done on Korean
- `synsemclass_prediction.py` — a wrapper for the SynSemClass classification model for easier use
- `parallel_corpus_write.py` — creates a preannotated evaluation parallel corpus from an unannotated parallel corpus as described in section 2.6
- `evaluation.py` — performs the experiments that measure the performance of the toolchain as described in section 3.1 using an annotated parallel corpus
- `aggregation_experiments.py` — performs the aggregation experiments described in section 3.2
- `csv_manipulator.py` — makes gold data for the aggregation experiments
- `synsemclass_writer.py` — writes the preannotated SynSemClass language-specific file
- `predictions.py` — a definition of a class used to hold the predictions of the toolchain
- `shared.py` — utilities shared by multiple scripts

[1]`https://github.com/petrkasp/synsemclass-pipeline`
[2]`https://aic.ufal.mff.cuni.cz/index.php/Main_Page`
[3]Original repository: `https://github.com/strakova/synsemclass_ml`

- `tmx_parser.py` — parses .tmx parallel corpus files[4]

B.1 Main toolchain

`main.py` takes several arguments as described in the user documentation (Appendix A), one of them being the task to perform. Each task follows a similar path:

First we locate verbs with UDPipe 2 (Straka, 2018). We use the UDPipe 2 REST service[5] and we perform the requests as in the official UDPipe 2 client[6]. At the time we were writing the initial implementation, UDPipe 2 locally was not readily supported and documented. For larger corpora, we send batches of up to 500 sentences. We then use the `ufal.udpipe` Python package to parse the CoNLL-U output of UDPipe 2. We have a custom `Verb` class that holds information on each verb. Here, we create instances of this class for each verb found by UDPipe 2. As the toolchain progresses, it fills these instances with more information on the verbs.

The next step is adding alignments with SimAlign (Jalili Sabet et al., 2020). We describe SimAlign in section 2.4. We give it the words as tokenized by UDPipe 2 and use the itermax matching algorithm — the custom matching algorithm introduced by Jalili Sabet et al. (2020) alongside the SimAlign algorithm.

The last common step is adding the predictions from the SynSemClass classification model. For this, we create a `DeferredPrediction` class that stores the inputs for the predictions inside the `Verb` instances. We then collect the deferred predictions and run them at once. We do this so that we can run the predictions in larger batches, instead of running each prediction separately.

The results are processed depending on the task chosen. For the parallel corpus creation task, the parallel corpus is written using the native Python library `xml.etree.cElementTree`. The SynSemClass preannotation file creation task also uses this library. The evaluation task compares the results generated by the toolchain to the gold data and calculates various metrics as described in section 3.1. The task that stores the results for later use in the aggregation experiments described in section 3.2 uses the native Python libraries `lzma` and `pickle`.

[4]We do not use any tmx corpora in our experiments, but we considered several OPUS corpora in this format.

[5]`https://lindat.mff.cuni.cz/services/udpipe/`

[6]`https://github.com/ufal/udpipe/blob/udpipe-2/udpipe2_client.py`

B.2 Aggregation experiments

For the aggregation experiments, we load the gold data as created by the `csv_manipulator.py` script and the predictions of the toolchain that were created and stored with the `pred-matrix` task. The aggregation script contains several argument presets for the correct pairings of the gold data and stored predictions. For our manually annotated Korean data, there are two variants — one filters the gold data to only contain the lemmas that were predicted by the toolchain, as is done with the rest of the languages, and the other variant keeps the gold data intact. The later variant is used in the experiments as described in section 3.2.

The `aggregation_experiments.py` script then goes through each setting of the aggregation parameters and produces the precision-recall curves for this setting. We use our custom implementation of the computation of the values for the precision-recall curves, as our setting is not strictly classification as described in section 3.1.1, so we cannot simply use standard implementations like the one in the Python library `scikit-learn`. For the plotting, we use the Python library `matplotlib`.

C. Parallel corpus format

The pipeline uses a custom XML format for parallel corpora with SynSemClass annotations. See listing 1 and listing 2 for an example.

The top tag is `sentences` and it is filled with individual `sentence` tags. The `sentence` tags have an `id` attribute that denotes the position of the sentence in the original corpus. Each `sentence` tag has three child tags: `text`, `source` and `verbs`.

The `text` tag contains the text of the sentence in the target language and the `source` tag contains the text of the sentence in the source language. They are followed by a `verbs` tag that contains instances of verbs in the target language in the form of `verb` tags.

Each `verb` tag contains a `class` attribute for the SynSemClass class that is not yet assigned by the pipeline and should be decided upon by the annotator. The SynSemClass class names are in the form of `vec` followed by 5 digits, e.g., `vec00017`. As we manually annotated the verbs into a constantly developing and as such, yet unfinished, class set of the SynSemClass ontology, we inevitably came across verbs without an assigned class in the ontology. In such cases, when we believed the verb sense is not yet annotated in the ontology, we inserted `TBD` instead of the class name. The `verb` tag also contains a `lemma` attribute generated by the pipeline.

Inside each `verb` tag, a `mark` tag has the sentence in the target language marked with the hat symbol (`^`) for direct use with the classification model. The generated corpus then contains a `predictions` tags with several `pred` tags for possible SynSemClass classes. Each `pred` tag has a `prob` attribute for the probability of this class in percentages, i.e., the max value is 100. The class is in the text of the tag. The `predictions` tag is followed by an `alignment` tag. If no alignment was made, the tag is empty, otherwise it contains the verb marked in the source sentence. In the generated corpus, if an alignment was made, an `alignment_predictions` tag follows with the same format as the `predictions` tag.

When we annotated the corpus, we fill in the missing alignments, but in some cases, no good alignment is possible. For this purpose, we add a `to` attribute to the `alignment` tag in the annotated corpus. In most cases, the `to` attribute is set as `verb`. This indicates that the verb on the target side has aligned with a verb on the source side. If no alignment is possible, we leave the `alingment` tag empty and set the `to` attribute to `none`. If some alignment is possible, but the aligned words have different meanings, we set the `to` attribute to the part of speech of the aligned word, or if it is a verb with a different meaning from the one on the target side, we set the `to` attribute to `verb-diff`.

Listing 1 An example parallel corpus as generated by the pipeline. For simplicity, the source language is English, but in CAPITAL letters. For demonstration, alignment on the second verb failed.

```
<sentences>
  <sentence id="0">
    <text>I sleep and eat.</text>
    <source>I SLEEP AND EAT.</source>
    <verbs>
      <verb class="" lemma="sleep">
        <mark>I ^ sleep and eat.</mark>
        <predictions>
          <pred prob="58.6">vec00735</pred>
          <pred prob="2.6">vec00440</pred>
          <pred prob="2.2">vec00921</pred>
          <pred prob="1.8">vec01118</pred>
          <pred prob="1.7">vec00556</pred>
        </predictions>
        <alignment>I ^ SLEEP AND EAT.</alignment>
        <alignment_predictions>
          <pred prob="82.5">vec00077</pred>
          <pred prob="7.4">vec00270</pred>
          <pred prob="3.8">vec00092</pred>
          <pred prob="1.7">vec00337</pred>
          <pred prob="1.6">vec00810</pred>
        </alignment_predictions>
      </verb>
      <verb class="" lemma="eat">
        <mark>I sleep and ^ eat.</mark>
        <predictions>
          <pred prob="46.9">vec00077</pred>
          <pred prob="13.0">vec00092</pred>
          <pred prob="4.4">vec00270</pred>
          <pred prob="1.3">vec00337</pred>
          <pred prob="1.2">vec00120</pred>
        </predictions>
        <!--Here the alignment failed.-->
        <alignment></alignment>
      </verb>
    </verbs>
  </sentence>
  <sentence id="1">
    <!--another sentence would be here-->
  </sentence>
  <!--more sentences would follow-->
</sentences>
```

Listing 2 The parallel corpus from listing 1 after it was annotated by an annotator.

```
<sentences>
  <sentence id="0">
    <text>I sleep and eat.</text>
    <source>I SLEEP AND EAT.</source>
    <verbs>
      <verb class="vec99999" lemma="sleep">
        <mark>I ^ sleep and eat.</mark>
        <alignment to="verb">I ^ SLEEP AND EAT.</alignment>
      </verb>
      <verb class="vec12345" lemma="eat">
        <mark>I sleep and ^ eat.</mark>
        <alignment to="verb">I SLEEP AND ^ EAT.</alignment>
      </verb>
    </verbs>
  </sentence>
  <sentence id="1">
    <!--another sentence would be here-->
  </sentence>
  <!--more sentences would follow-->
</sentences>
```