ChatGPT

Form Field Alignment and Checkbox Mapping for Retirement Plan Adoption Agreements

Overview of the Alignment Challenge

Retirement plan *Adoption Agreements (AAs)* are complex legal forms that present electable plan provisions (options selected via checkboxes or fill-in fields) in varying layouts depending on the vendor (e.g. Relius vs. Ascensus). Aligning these fields across different templates is challenging because each vendor may structure and phrase the same provision differently. Key difficulties include **provenance drift** (semantic matchers getting confused by similar text in the wrong section), inconsistent numbering and formatting, and one-to-many mappings where a single conceptual choice in one template corresponds to multiple fields in another. Moreover, in a compliance context, **precision is paramount** – a false positive mapping (mis-aligning two different provisions) is riskier than missing a match, since it could lead to incorrect plan configurations. Below, we survey best practices for extracting form fields and checkboxes from legal PDFs, representing question–option text for matching, using hybrid retrieval (structure + embeddings), aligning fields with high precision (including one-to-many cases), evaluating alignment accuracy, and leveraging domain-tuned embeddings for improved semantic matching.

Robust Extraction of Form Fields and Checkboxes

Accurate field alignment starts with reliable **extraction of form fields and their labels** from the PDF documents. In practice, OCR and layout-parsing tools like Azure's Document Intelligence (formerly Form Recognizer) can detect **selection marks (checkboxes)** and text, but often do **not automatically link a checkbox to its descriptive label** [1]. This means the raw output might tell you a checkbox's coordinates and whether it's checked, but not which question or option it corresponds to. Two complementary strategies can address this:

- **Spatial Proximity Matching:** Extract all checkbox marks and text blocks on the page, then match each checkbox to the nearest relevant text by coordinates [1]. In many forms, the label text is immediately to the right of a checkbox or just above/below it. A simple heuristic is to search within a certain pixel radius (e.g. 50–100px) around each checkbox's bounding box for text [2]. This captures most label–checkbox pairs. Be mindful of reading order: the closest text in raw coordinates might occasionally be a neighboring field if the layout is dense or skewed. It's wise to incorporate slight tolerance for misalignments in scanned documents (which can have minor rotations or shifts) [3].

- **Leverage Table Structures:** Many adoption agreement forms use tables or multi-column layouts to organize options. If the OCR/AI can detect table structure, use that to your advantage. Azure's form analysis often identifies table cells; when checkboxes are inside table cells, the cell's text will include the label [4]. By extracting the table, you inherently get each checkbox **grouped with its label in the same cell** [4]. This approach tends to be more reliable than purely distance-based matching, because it respects the form's intended structure. Always inspect the model's table output – in welldesigned forms, each row might correspond to a question, with cells for the option text and a checkbox indicator. Using table context can drastically reduce errors in label association [5].

In cases where the prebuilt models still struggle (for example, unusual layouts or very dense text), consider fine-tuning a custom model. Azure Document Intelligence allows labeling fields manually to train a custom form extractor [6]. With a few labeled examples of a particular vendor's AA, the model can learn associations (e.g. that a checkbox in column X corresponds to the text in column Y on the same row). But even with custom models, it's prudent to apply the above **post-processing checks** (spatial matching or table validation) to catch any missed links.

Representing Question–Option Pairs for Embedding

Once fields and their labels are extracted, the next step is to create a representation suitable for semantic matching. Each electable provision often consists of a **question or prompt plus one or more options/answers** (e.g. *"Eligibility Waiting Period:"* with checkboxes for *Immediate, 3 months, 6 months,* etc.). For embedding-based similarity, treat the **question-and-option text as a combined unit** so that the model captures their joint meaning. A common technique, inspired by QA retrieval tasks, is to concatenate the prompt and a given option into one string before encoding. For example, one could format the text as: *"Eligibility Waiting Period – Option: 6 months"* and compute a single embedding for that pair. In transformer-based models, this is akin to feeding *[CLS] question [SEP] option [SEP]* and using the resulting [CLS] embedding as the field representation [7]. By including both the context (question) and the specific option text, we ensure the embedding encodes the full semantic of *that particular choice*. This is especially important if a single question has multiple checkboxes – each option should be represented distinctly, since "Immediate eligibility" vs "6-month eligibility" need to map to different concepts.

Preprocessing to remove noise is critical before embedding. Legal forms are full of numbering, lettering, and formatting artifacts (e.g. *"3.1(a) ..."* or *"Section IV – 2"*). These identifiers are useful for humans navigating the document, but they can mislead an embedding model by introducing irrelevant tokens. Best practice is to **strip out clause numbers, bullets, and other non-semantic markers** from the text used for embeddings. For instance, *"3.1(a) Eligibility of Employees – [ ] Immediate"* should become *"Eligibility of Employees – Immediate"*. This focuses the vector on the substantive content rather than superficial overlaps in numbering. Reducing such "structural noise" helps the model measure true semantic similarity instead of picking up, say, two fields both labeled "Section 3" (which would be a meaningless coincidence). As a general rule, any consistent patterns that are purely structural (like "Article I:" at the start of section titles) can be removed or down-weighted in the text representation [8] [9]. By cleaning typos and extraneous symbols, you mitigate random variations that can distort embedding similarities [8] [10].

Additionally, consider the **granularity** of what you embed. If a provision consists of a question plus multiple fillable blanks (for example, a sentence with two blanks to be filled in), you may choose to embed the entire sentence as one chunk (so the model sees the context of both blanks together). On the other hand, if each blank corresponds to a distinct parameter (e.g. one blank for a percentage and another for a number of years, under the same question), it might be better to treat them as separate fields linked to the same question context. In practice, one can experiment: either concatenate all related fields into one text (e.g. *"Loans: maximum % __;* ***minimum age*** *____"*) or embed them separately but include the shared prompt text in each. The goal is to create representations that can be compared across templates even if the structure differs (one template might have those two blanks as one combined sentence, another might break them into two questions). Maintaining the *semantic unit* of each electable choice is key for alignment.

Hybrid Retrieval with Structure and Embeddings

To improve mapping accuracy, use a **hybrid retrieval approach** that combines structural cues (metadata filters) with semantic embedding similarity. In other words, don't rely on raw embeddings alone across the entire documents; first **narrow the candidate pool using document structure**, then apply vector similarity ranking within that subset [11] [12]. This mitigates the risk of "provenance drift," where an embedding might erroneously match a similar-sounding text from the wrong section. Practical steps include:

- **Metadata Filtering by Section:** Leverage the fact that adoption agreements are organized by article/section (e.g. "Eligibility," "Vesting," "Distributions"). If you can detect or label these sections, tag each field with its section name or a section ID. At query time (i.e. when finding a match for a given field), restrict the search to fields in the **same section or a logically equivalent section** of the other document [11]. For example, if we're matching a field under "Loans" in Template A, it should primarily be compared to fields under "Loans" in Template B or the canonical schema, rather than scanning the entire document. This structural filter acts as a high-precision gate: it cuts off many spurious matches that happen to share keywords but are about different topics. In practice, you might allow a bit of flexibility (maybe sections that are closely related can cross-match), but generally, aligning sections first then fields within them yields a big precision boost.

- **Candidate Generation with Lexical + Semantic Search:** Within the filtered subset of potential matches, use a combination of **lexical matching and dense embeddings** to retrieve candidates. A *sparse lexical search* (like BM25 or keyword overlap) can catch obvious terminology matches (e.g. both forms mention "hardship withdrawals"), which helps recall for cases where synonyms aren't used. Dense embeddings will capture rephrased semantics (e.g. *"termination of employment"* vs *"severance from service"*). A **hybrid search** that merges results from BM25 and embeddings often outperforms either method alone [12]. The dense model might retrieve a few semantically close chunks that lack exact keyword overlap, while BM25 ensures you don't miss fields that use identical phrasing. By unioning these and then re-scoring, you get a robust set of candidates for each field [13]. Many modern systems implement this by performing two searches and taking the top K from each ("two-tower" approach), or by indexing documents with both types of vectors.

- **Cross-Encoder Re-ranking for Precision:** After obtaining top candidates via the fast retrieval stage, apply a more precise *cross-encoder* or verification model to re-rank or filter them [11] [12]. A cross-encoder (e.g. a BERT model fed the pair of field texts together) can make a **fine-grained judgment** about whether a candidate truly matches the query field, often catching subtle differences that embeddings (which encode each item independently) might overlook. For instance, two fields might both mention "eligible employees," but one is about *eligibility for employer contributions* and another about *eligibility for plan entry* – a cross-encoder can read both texts in full and notice the context mismatch, down-ranking that pair. In one recent contract clause retrieval benchmark, a pipeline of a bi-encoder retriever followed by an LLM re-ranker substantially improved the relevance of top results [14]. When high precision matters (as it does in compliance mappings), this reranking step is invaluable [11]. It's acceptable to sacrifice a bit of recall here – by discarding any low-scoring pairs – because *false positives are far more harmful* in this domain than false negatives. The end result is a short list of candidate mappings that are both semantically and contextually aligned. These can either be auto-accepted or passed to human reviewers for final confirmation, depending on your tolerance for risk.

Aligning Electable Fields Across Different Templates

With extracted data and a hybrid retrieval mechanism in place, the core task is to **map each electable provision from one template to its counterpart in another** (or to a canonical schema). This process must handle one-to-one matches as well as one-to-many or many-to-one scenarios.

Use a Canonical Schema as an Intermediary: A best practice in multi-template alignment is to define a **canonical set of provisions** (a superset of all possible options across vendors) and map each template's fields to this schema. By mapping Template A and Template B both to the canonical, you indirectly achieve A-to-B alignment while naturally handling one-to-many cases – a single canonical item might link to multiple fields in a given template. For example, suppose the canonical schema has an element "Automatic Enrollment Percentage". Vendor X's AA has one field for this (the percentage blank), but Vendor Y's AA splits it into two fields (one checkbox to indicate auto-enrollment is adopted, and a blank for the percentage). Both of Vendor Y's fields would map to the one canonical concept. Structurally, you'd group them as a compound mapping. Using the canonical as a hub ensures that each vendor's idiosyncratic breakdown still converges on the same *conceptual provision*.

One-to-Many and Many-to-One Alignment Strategies: When aligning fields, identify cases where a **single field in one document corresponds to multiple fields in another**. This often happens if one form asks a combined question that another form breaks into parts. Handle this by either **concatenating the multiple fields** on the side that has many (creating a combined text that can be compared as one unit), or by mapping all of them to the same canonical group. For instance, one template might ask *"Should Roth contributions be allowed? If yes, what is the maximum percentage?"* in one question. Another template might have a yes/no checkbox for Roth, and a separate field for the percentage. In alignment, you'd map both the yes/no and the percentage from Template B to the single combined question from Template A. Embeddings alone might not realize these two are connected, so you may need a rule or a custom logic: if the text of one field is a subset or complement of another, they likely form an aligned set rather than separate concepts. During candidate matching, a field that only makes sense in context (e.g. a blank percentage with no question text) should *carry the context from its sibling field*. This can be done by augmenting the blank field's text with the prompt of its parent question before embedding, so that "_ **%" becomes "Roth max percentage:** ___%" internally.

Avoiding Structural Traps: Structural traps refer to being misled by form structure when aligning. For example, some forms include *placeholder lines or blank tables* for "write-in" provisions – these might look like fields but actually aren't applicable (left blank). A naive matcher could attempt to align a non-existent choice. To avoid this, implement high-precision rules such as: *if a field has no label text or is blank, do not attempt to map it*. Also consider the hierarchy: if certain fields only apply when a preceding field is "Yes," you shouldn't align an optional sub-choice out of context. It's helpful to propagate conditional context as part of the field description (e.g. include "**[If loans are allowed]**" in the text when embedding a loan sub-option so it doesn't match something unrelated).

Finally, set **conservative matching thresholds** and use human-in-the-loop for verification on critical mappings. It's better to miss a mapping (which can be later manually linked) than to erroneously link two different provisions. In practice, this might mean requiring an embedding similarity score well above a tuned cutoff *and* requiring section metadata to match, before declaring an automatic alignment. Any potential match that falls below that strict threshold can be flagged for manual review. This aligns with the philosophy observed in legal document retrieval benchmarks, where presenting only the top highly-relevant results is preferred – users can be misled by seeing moderately relevant but imprecise suggestions [15] [16]. In our context, that means our system should output a mapping only when it's very confident (high semantic score and structural consistency); otherwise, it should stay silent (or say "no match found") rather than guess. Such a high-precision bias inevitably lowers recall, but given the compliance risk, it's an appropriate trade-off.

To further boost precision, an emerging approach is to employ an LLM to do a final sanity check. For example, you could prompt GPT-4 with: *"Do these two provisions essentially mean the same thing in the context of a 401(k) plan? Text A: ... Text B: ..."*. The model's answer could be used as a last gate (though be cautious – LLMs can sometimes be inconsistent or require careful prompt calibration). Early experiments in contract clause alignment show that large models can capture subtle differences that simpler algorithms miss [17] [18], but they should be used to complement, not replace, the deterministic methods described.

Evaluation Methods for Field Alignment

Evaluating semantic alignment in this domain requires ground truth mapping between forms, typically curated by domain experts (ERISA consultants or plan compliance specialists). Each field in each template would be labeled with the canonical provision it corresponds to (or marked as "no mapping" if it's a vendor-specific extra). From this reference alignment, standard **information retrieval metrics** can be computed to assess the AI mapper's performance [19]:

- **Precision:** The fraction of field mappings proposed by the system that are *correct*. A mapping counts as correct if it truly links equivalent provisions. High precision means when the system says two fields match, they almost certainly do. Formally, $Precision = TP/(TP + FP)$ where *TP* is the number of true-positive matches the system found and *FP* is false positives [19]. In our scenario, a false positive might be aligning a "Beneficiary Designation" field to an unrelated "Benefit Formula" field – clearly wrong and would count against precision.

- **Recall:** The fraction of true alignments that the system successfully identified. $Recall = TP/(TP + FN)$ [20], where *FN* are the ground-truth mappings the system missed. If a particular provision exists in both Template A and B but the system failed to match them, that's a false negative. High recall means the system found most of the real equivalences. However, as noted, we are often willing to sacrifice some recall to keep precision high.

- **F1 Score:** The harmonic mean of precision and recall [19]. This gives a single-score balance of the two. If comparing different matching approaches or doing hyperparameter tuning (e.g. adjusting the similarity threshold), F1 can serve as an overall indicator. But in a compliance setting, you might monitor precision more strictly – for instance, ensure precision stays above, say, 95%, while trying to maximize recall under that constraint.

For alignment tasks, evaluation can also be done at different levels of strictness. One might consider *partial credit* in one-to-many alignments. For example, if a canonical provision maps to two fields in Template B, and the system only matched one of them, is that a half-correct or fully wrong? Depending on needs, you could count that as a true positive (since it did find a correct link) but still note that something was missed. Alternatively, structure the evaluation such that each canonical-item to vendor-field mapping is an atomic pair to be found.

Another useful metric is **Top-1 accuracy** if the system ranks multiple possible matches. If you allow the system to suggest, say, the best match and a second-best match for each field, you could measure success@1 and success@K. In our case, we typically want one correct mapping, so Top-1 accuracy essentially corresponds to precision (assuming the system always gives one answer per field). If a ranking approach is used (like listing the top 3 candidate matches for a field), metrics like Mean Reciprocal Rank (MRR) or Normalized Discounted Cumulative Gain (NDCG) could be calculated. However, these are more common in search applications. In form alignment, it's more binary: did you map it correctly or not. Notably, researchers caution that moderate relevance is not good enough – for example, in clause retrieval, a system that returns several "okay" matches is less useful than one that returns a few *highly relevant* ones [15]. Similarly, our evaluation should reward **precision at the highest relevance**. A practical approach is to count an alignment as correct only if it's the exact intended counterpart, and anything else is wrong. This yields clear precision/recall measurements that align with compliance requirements (either the plan provision was correctly mapped, or it wasn't – there's little room for "almost correct" in a regulated context).

Human review and qualitative evaluation are also important. Because errors can be costly, many teams do a manual audit of a sample of mappings, especially any that were automatically inferred with lower confidence. Field-by-field review by a subject matter expert can catch subtle mismatches that metrics might not reveal. For instance, two fields might be mapped and share keywords, but upon reading, an expert realizes one has an extra condition that the other doesn't. Such insights can be fed back as new rules or training data for the system. Over time, building a **confusion matrix** of commonly mistaken pairs can help refine the alignment logic (e.g., the system often confuses "matching contributions" vs "employee contributions" because of shared words – so add logic to distinguish those by context).

Domain-Specific Embeddings and Fine-Tuning

General-purpose language models and embeddings (like those from BERT, RoBERTa, or OpenAI) provide a strong starting point, but they may not fully grasp the intricacies of legal plan documents. Adopting **domain-specific embeddings** can significantly improve semantic alignment in this context [21] [22]. There are two main approaches: using a pre-trained legal/domain model, or fine-tuning a model on your specific corpus.

Leverage Legal Pre-trained Models: Researchers have developed transformer models like *LegalBERT* (trained on legal texts such as contracts and case law) and *CaseLaw-BERT* (focused on judicial opinions). These models have vocabulary and representations more attuned to legal language. For example, a general BERT might not intrinsically know that "hardship withdrawal" is related to retirement plans, but a model exposed to lots of plan documents or legal text might. By using such a model as your embedding generator, you get vectors that likely position legally and semantically similar phrases closer together. In one case, an AI system for contracts found that a fine-tuned model could better differentiate subtle legal terms – *e.g.* understanding that "consideration" in a contract is not about being polite, but a legal term [23] [21]. Domain models reduce misunderstandings of jargon and thus reduce false positives in similarity matching [22].

Fine-Tune on Adoption Agreement Data: The ultimate specialization is to fine-tune an embedding model on a dataset of adoption agreements or Q&A pairs derived from them. If you have a large number of example forms (possibly annotated with canonical alignments), you can train a model (for example, fine-tune a Sentence-BERT or similar bi-encoder) to directly learn the notion of "equivalent provisions." During fine-tuning, you would feed pairs of text: positive pairs (field from Template A aligned with field from Template B) and negative pairs (field from A with an unrelated field from B), so the model learns to embed aligned fields closer and non-matches farther apart. This kind of supervised fine-tuning can be powerful: it trains the model on the *actual task* of semantic field alignment. Even if you don't have many labeled pairs, you could leverage unlabeled data through self-supervision: for example, take one template, perturb the text slightly (or use paraphrasing) and train the model that the original and perturbed are a match (positive pair), whereas random pairings are not. Fine-tuning must be done carefully to avoid overfitting, but with proper validation it can significantly boost alignment accuracy in niche domains [21] .

Capturing Nuanced Context: In retirement plan documents, small phrasing differences can carry big implications (consider *"shall"* vs *"may"*, or *"maximum 10%"* vs *"exact 10%"*). A domain-tuned model is more likely to catch these nuances. For example, a generic model might vectorize "the employer **shall** make matching contributions" and "the employer **may** make matching contributions" as very similar (because most words overlap), potentially suggesting a false alignment. A model trained on legal distinctions might recognize the modality difference and treat them as different meanings. Similarly, domain-specific training helps with multi-word terms of art (like "Top-Heavy Minimum Contribution" or "ADP Test") that a general model might not cluster correctly. Fine-tuning effectively *teaches* the embedding space to reflect the domain's concept structure – terms that lawyers consider synonymous end up nearby in the vector space, and those that differ in meaning are separated [22] .

Evaluation of Domain Embeddings: When adopting a specialized model, always evaluate it on some known alignments or semantic similarity tests in your domain. Compare it against a baseline model using the metrics above (precision/recall on a set of mappings). Often, fine-tuning yields a noticeable jump in both recall and precision because the model stops being "distracted" by general-language correlations and focuses on what matters in plan provisions [22] . For instance, after fine-tuning on a legal corpus, an embedding model might no longer mistakenly group "trustee" with "trust" (different concepts) and instead group it with "plan administrator" if that's more contextually correct. If direct fine-tuning is not feasible, another approach is **prompting a large language model** to act as an embedder for legal text. With appropriate prompts, models like GPT-4 can generate embeddings or similarity judgments that respect legal context – essentially a form of "on-the-fly domain adaptation" via prompting. This is an active area of experimentation and might not be necessary if a fine-tuned static model does the job.

In summary, investing in domain-specific embeddings – whether via pre-trained legal models or custom fine-tuning – is a proven way to improve semantic matching for legal forms. It **sharpens the system's understanding of jargon and subtle differences**, thereby **reducing false positives and improving confidence** in the alignment [22] . Combined with the structural and retrieval techniques discussed, this forms a robust approach to mapping adoption agreement fields with the high precision that legal compliance work demands.

Conclusion

Mapping form fields and checkbox selections across different adoption agreement templates is a complex task at the intersection of document AI and legal knowledge. The key is to **preserve context** at every step: from correctly linking checkboxes to their labels, to representing each question-option pair in a meaningful way for similarity comparison, to using the surrounding structure as a guide for alignment. By using a **hybrid approach – pairing layout-based rules and metadata with powerful embeddings – and layering in domain expertise through fine-tuned models and careful validation**, one can achieve high-precision alignments. Evaluation should be continuous and stringent, focusing on eliminating false matches and confirming that every mapped pair truly corresponds to the same legal choice. Adopting these best practices, teams have reported substantial improvements in alignment accuracy, turning what used to be a manual, error-prone process into a streamlined (partly automated) pipeline with *AI-assisted mapping* of retirement plan provisions. The result is better consistency across plan documents, time saved for analysts, and ultimately more reliable compliance configurations. As both language AI and document understanding technology evolve (e.g. layout-aware transformers, improved OCR for checkboxes), we can expect even more robust solutions to emerge for form-field alignment in the legal domain. For now, the combination of **robust extraction, smart text representation, structural filtering, semantic matching, and domain calibration** offers a state-of-the-art toolkit for tackling the challenges described.

Sources:

- Azure Document Intelligence community insights on checkbox extraction [1] [4]
- Embedding combined question-option text for semantic similarity [7]
- Hybrid retrieval and reranking best practices (semantic + structural filtering) [11] [12]
- Ontology alignment metrics adapted for field mapping (Precision/Recall definitions) [19]
- Emphasis on high-precision results in legal IR (clause retrieval study) [15] [16]
- Benefits of domain-specific and fine-tuned embeddings for legal text [21] [22]

---

[1] [2] [3] [4] [5] How can I extract checkbox labels from PDF forms using Azure Document Intelligence API? - ai employee - Latenode Official Community
https://community.latenode.com/t/how-can-i-extract-checkbox-labels-from-pdf-forms-using-azure-document-intelligence-api/31783

[6] How to extract checkbox answer as value and question as key? - Microsoft Q&A
https://learn.microsoft.com/en-us/answers/questions/1662379/how-to-extract-checkbox-answer-as-value-and-questi

[7] arxiv.org
https://arxiv.org/pdf/2212.10558

[8] [9] [10] How does noise affect similarity calculations in embeddings?
https://milvus.io/ai-quick-reference/how-does-noise-affect-similarity-calculations-in-embeddings

[11] [12] [13] Optimizing Chunking, Embedding, and Vectorization for Retrieval-Augmented Generation | by Adnan Masood, PhD. | Medium
https://medium.com/@adnanmasood/optimizing-chunking-embedding-and-vectorization-for-retrieval-augmented-generation-ea3b083b68f7

[14] [15] [16] [17] [18] aclanthology.org
https://aclanthology.org/2025.acl-long.1206.pdf

[19] [20] Performance assessment of ontology matching systems for FAIR data | Journal of Biomedical Semantics
https://link.springer.com/article/10.1186/s13326-022-00273-5

[21] [22] [23] Should I fine-tune an embedding model for a specific area of law?
https://milvus.io/ai-quick-reference/should-i-finetune-an-embedding-model-for-a-specific-area-of-law